ED 016 483 JC 671 001

COMMENTS AND CURRICULA RELATED TO THIRD GENERATION DATA PROCESSING AND SPECIFICALLY IBM SYSTEM/360.

BY- CLARK, JOHN R. TILTON, THEODORE
ORANGE COAST COLL., COSTA MESA, CALIF.
CALIFORNIA STATE DEPT. OF EDUCATION, SACRAMENTO

PUB DATE OCT 67
EDRS PRICE MF-$1.00 HC-$8.84 219P.

DESCRIPTORS- *JUNIOR COLLEGES, *DATA PROCESSING, *COMPUTERS, *VOCATIONAL EDUCATION, *TECHNICAL OCCUPATIONS, CURRICULUM PLANNING, SUBPROFESSIONALS, COBOL, PL/1, FORTRAN,

TWO INSTRUCTORS ATTENDED CLASSES AT THE IBM EDUCATION CENTER TO FIND OUT HOW THE COURSES IN THIRD-GENERATION COMPUTER HARDWARE AND OPERATING SYSTEMS SHOULD BE TAUGHT AT THE JUNIOR COLLEGE LEVEL. THEY LEARNED THE TEACHING TECHNIQUES AND WERE ABLE TO SELECT THE ESSENTIAL ELEMENTS FOR THE CURRICULUM. THE SUGGESTED 4-SEMESTER PROGRAM CONTAINS AN INTRODUCTION TO DATA PROCESSING, A STUDY OF THE REPORT PROGRAM GENERATOR, PROCEDURAL LANGUAGES (COBOL AND PL/1), AND ASSEMBLY LANGUAGE CODING, WITH LIMITED ELECTIVES (FORTRAN IV AND UNIT RECORD EQUIPMENT), AND A POST-CERTIFICATE COURSE IN ADVANCED SYSTEM OPERATIONS. THE PROGRAM WILL PROVIDE THE STUDENT WITH A GENERAL BACKGROUND (EVOLUTION, TERMINOLOGY, AND TECHNIQUES OF DATA PROCESSING) AND ORIENT HIM TO THE MANY FACETS OF THE FIELD. EVEN AT THE END OF THE FIRST YEAR, THE STUDENTS WILL BE EMPLOYABLE, WHILE THOSE WHO COMPLETE THE COURSE WILL HAVE THE CONCEPTS OF GREATEST VALUE TO A PROSPECTIVE EMPLOYER. THE IMPACT OF THIRD-GENERATION COMPUTERS ON EXISTING DATA PROCESSING CURRICULA, PREPARATION FOR THE INEVITABLE FOURTH GENERATION, THE TRAINING OF INSTRUCTORS, AND OTHER PERTINENT TOPICS ARE COVERED. APPENDIXES INCLUDE COURSE OUTLINES AND PROGRAMING EXAMPLES. (HH)

ERIC

ED015483

COMMENTS AND CURRICULA RELATED TO THIRD GENERATION DATA PROCESSING AND SPECIFICALLY IBM SYSTEM/360

U.S. DEPARTMENT OF HEALTH, EDUCATION & WELFARE
OFFICE OF EDUCATION

THIS DOCUMENT HAS BEEN REPRODUCED EXACTLY AS RECEIVED FROM THE PERSON OR ORGANIZATION ORIGINATING IT. POINTS OF VIEW OR OPINIONS STATED DO NOT NECESSARILY REPRESENT OFFICIAL OFFICE OF EDUCATION POSITION OR POLICY.

UNIVERSITY OF CALIF.
LOS ANGELES

DEC 6 1967

CLEARINGHOUSE FOR
JUNIOR COLLEGE
INFORMATION

AN IN-SERVICE TRAINING REPORT
DEVELOPED AT ORANGE COAST COLLEGE
IN COOPERATION WITH
THE STATE DEPARTMENT OF EDUCATION

BY

JOHN R. CLARK
ASSOCIATE PROFESSOR
DATA PROCESSING/MATHEMATICS

THEODORE TILTON
ASSOCIATE PROFESSOR
DATA PROCESSING/MATHEMATICS

OCTOBER 1967

JC 67/001

ACKNOWLEDGEMENT

This report is the result of attendance at IBM customer classes on System/360 on an in-service training basis during the 1966-67 school year by John R. Clark and Theodore Tilton. Impetus to this study was generated by the advent of third generation technology and the recognition by the State Department of Education and Orange Coast College that immediate steps were needed if an effective occupational curriculum was to be maintained and instructional skills upgraded to meet the challenge of transition from second to third generation technology.

Support in development of this report and in implementation of this program was given by:

California State Department of Education
Bureau of Junior College Vocational-Technical Education

Mr. Leland P. Baldwin, Chief
Mr. Philip E. DeMarco, Consultant in Business Education

Orange Coast College Junior College District

Dr. Norman E. Watson, District Superintendent
Dr. Robert B. Moore, President, Orange Coast College
Dr. James F. Fitzgerald, Dean of Instruction, O.C.C.
Mr. Gilbert D. Saunders, Administrative Assistant
Mr. Bernard J. Luskin, Assoc. Dean, Federal Projects
Mr. Donald E. McKee, Dir. Educ. Data Proc., O.C.C. '67-
Mr. Richard W. Brightman, Dir. Educ. Data Proc., O.C.C. '65-7
Mr. Charles H. Lewis, Chairman, Math and Sci. Division, O.C.C.
Mr. Charles G. Haley, Chairman, Business Division, O.C.C.

International Business Machines Corporation

Mr. Andrew A. Burgoyne, Associate Account Representative,
Education.

TABLE OF CONTENTS

BACKGROUND 1

ANSWERS TO QUESTIONS 5

SUGGESTED THIRD GENERATION CURRICULUM 11

UNSOLVED PROBLEMS 24

THE DATA PROCESSING PROGRAM AT ORANGE COAST COLLEGE, 1967-68 35

APPENDIX

- SUGGESTED COURSE OUTLINES
  - REPORT PROGRAM GENERATOR
  - PROGRAMMING LANGUAGE I
  - ASSEMBLY LANGUAGE CODING

- PROGRAMMING EXAMPLES AND CLASS ROOM HANDOUTS
  - REPORT PROGRAM GENERATOR

- PROGRAMMING EXAMPLES AND CLASS ROOM HANDOUTS
  - PROGRAMMING LANGUAGE I

- PROGRAMMING EXAMPLES AND CLASSROOM HANDOUTS
  - ASSEMBLY LANGUAGE CODING
    - COMMERCIAL INSTRUCTION
    - GENERAL PURPOSE REGISTERS
    - FLOATING POINT
    - IOCS CARD & PRINTER FILES
    - IOCS TAPE FILES
    - IOCS DIRECT ACCESS FILES
    - IOCS INDEX SEQUENTIAL FILES
    - GENERAL MACHINE PROBLEM W/SOLUTIONS

BACKGROUND

With the announcement of third generation computers in the Spring of 1964 most junior colleges teaching data processing and those considering the implementation of vocational training in this area were faced with the problem of modifying their curricula for the impact of this new development. By 1965 the commercial acceptance of the third generation systems was evident. Today there seems to be no reversing the tide of third generation systems entering the commercial world. It is generally agreed that by 1970 the vast majority of data processing systems dealing with commercial, industrial, governmental and scientific applications will be of the third generation variety.

At present over 75% of all computers on order are of the third generation and over two-thirds of these are the IBM System/360. A leading investment service estimates 12,000 of these 360 units will be delivered in 1967 making the estimated year end total 17,850 units.

Acceptance of the third generation computer by the data processing community of Southern California was evident soon after its announcement. Since the major purpose of the data processing program is to provide instruction which will enable its graduates to obtain positions and advance in their chosen vocational fields it became quite evident that third generation training must be integrated into the curriculum. The administration and data processing department of Orange Coast College realized major modifications of its vocational program would be required.

if they were to serve the needs of the students and the community.

Realization of the inevitable modification of the educational program in data processing before its current shift of emphasis from first to second generation equipment could be completed was indeed a shock. More than anything it pointed out the dynamic features of this occupational area. Many questions were asked by instructors who were currently implementing a change from unit record and LGP-30 equipment to the IBM 1401 and 1620 Disk Oriented System. Among the many questions asked were the following:

1. Is the third generation computer system within the comprehension of the junior college student?
2. Can adequate instruction be given on the third generation computer and its operating system during a two year program without overloading the student?
3. What priority should be placed on the various phases of third generation languages and equipment?
4. How may the new material be presented in a standard three hour per week lecture format?
5. What effect will the third generation computer systems have on attitudes to hands-on training?

By late 1965 the potential of the third generation computers was becoming known. Claims of achievement and rumors of difficulties were rampant. Generalized information was available, but specific answers to questions of an educational nature were almost non-existent. It was decided that the best way to evaluate a third generation educational program would be to have present instructors undertake a study of the hardware and the operating systems to determine if and how they may be taught at the junior college level. This report is a partial culmination of this study.

Two instructors with a combined 14 years of work experience with computers and 6 years teaching data processing at the junior college level were selected for the study. Release time for the study was obtained by assigning the instructors as team teachers. By alternating on the preparations and lectures one instructor was available to pursue the evaluation of the new systems. Special arrangements were made with the Educational Division of International Business Machines for the scheduling of customer classes on third generation equipment.

Approximately 100 man days were spent in formal training at the IBM Education Center, Los Angeles. The courses listed below were taken by one or both of the instructors. With the exception of COBOL and FORTRAN IV, these courses were taken during the academic year 1966-67.

Number	Course Title	Length
S3599	System/360 Introduction	4 Days
P3631	System/360 Assembly Language	10 Days
P3637	System/360 Basic Operating System	10 Days
P3640	Report Program Generator	4 Days
P3638	Operating System/360 Coding	10 Days
P3693	Workshop BOS 16K Programming	5 Days
P3694	Workshop Operating System/360 Programming	5 Days
P3670	PL/I	5 Days
I2840	Graphic Data Processing	2 Days
P2840	Graphic Programming	3 Days
P3650	COBOL	5 Days
N0080	FORTRAN IV	4 Days

Courses presented at the Educational Center were conducted in a most professional manner. Instruction was well organized and excellent. Laboratory exercises were interspersed with the course material and the students were present when most of their problems were executed. The instructors not only received an understanding and an appreciation for the third generation computer and its software, but also obtained many teaching techniques and the materials, as shown in the appendix of this report.

One of the major benefits obtained from the training at the IBM Education Center was the contacts and discussions with fellow students. The majority of these men were supervisors or lead programmers for leading industries which were expecting to take or had just taken delivery of a third generation system. Advantage was taken of this opportunity to discuss the needs and expectations of the data processing industry. The results of these discussions are reflected in the choice, emphasis, and order of topics presented in the suggested curriculum.

ANSWERS TO QUESTIONS

As a result of attending the IBM System/360 classes and independent study, the instructors have formulated answers to the questions initially raised and stated in the previous section.

1. Is the third generation computer system within the comprehension of the junior college student?

There is no question in the minds of the instructors that the junior college student is capable of learning and achieving a level of mastery of the third generation equipment and system thereby enabling him to enter the employment market and achieve success in the data processing industry.

In terms of hardware design and execution speeds there is a tremendous advance in the third generation computer. But in the areas of programming languages and commercial applications in which the students will be working, the level of sophistication is approximately the same. Improvements in software will counterbalance many of the problems arising from new computer applications.

Students will not have trouble with interference from concepts related to second generation computers as have many professional programmers. Items such as data formats, job control cards, etc may be introduced with sugar coating as the student is learning the art of programming.

One of the initial conclusions obtained by both instructors after four weeks of assembly language and operating system study was: "This third generation is just a big 1620!" This conclusion is not valid. However, there are many similarities which may be shown between the 1620 Disk System and System/360.

2. Can adequate instruction be given on the third generation computer and its operating system during a two year program without overloading the student?

The answer to this question is somewhat qualified. Computer science today is so productive that it is a full time job to keep abreast of new developments. It has been stated that each installation should have a full time employee devoted to maintaining the operating system. Complete mastery of all the details of a third generation system is in all probability impossible in a lifetime since this is a dynamic, growing system. The degree of mastery obtained in a two year study of the third generation systems will not be as great as that obtained in the same time with unit record equipment (or second generation computers). Therefore, the guiding principle of the educational program will have to be:

1. Train the student so that he is competitive in the employment market and productive on his first job.
2. Provide the student with a foundation which will enable him to learn whatever new system concepts or generation computer he will have the opportunity to meet in his vocational career.

Two years of training is adequate to meet the conditions above. The curriculum we have outlined is not completed in two years. However, with minor modifications, it could be. We feel the prerequisite of full time employment for the post certificate course will improve the course and will impress upon all students the need for continuing education in the field of data processing.

3. What priority should be placed in the various phases of third generation languages and equipment?

The question of priority of languages appears to be dependent on two factors: time and the immediate employment area. Time is essential due to the fact that the student entering the program this fall will enter the employment market two years hence and reach his peak potential in five to eight years. Considering the fact that eight years ago very few junior college programs were established, the question of priority is important and will never be completely answered. The priority of instruction is also clouded with the claims and publications of many groups and committees who are creating or sponsoring "The Language" or system which will replace all others. While many of these claims may have some validity and the languages should be included in the curricula, the proper prospective can only be obtained by maintaining liaison with the employment community around the college.

From discussions with manufacturers, representatives, and members of industry in our area,we believe the most important programming systems in the immediate future will be the machine oriented S/360 Assembly Language and the problem oriented S/360 Report Program Generator. Thorough competence in these programming systems is readily becoming the most important technical prerequisite of the professional data processor.

With the advent of improved software accompanying the third generation computer, the question of priority of equipment is somewhat lessened. The utilization of new equipment or its modification has been reduced in many cases to changing of a few control cards or instructions in a given program. At the assembly language level, the programmer provides parameters for I/O Macros at the beginning of the program referencing a particular device and is relieved of writing the necessary commands to control the device. Many hardware devices may be controlled by the use of higher level languages, (e.g. Cathode Ray Display Stations may be effectively programmed using a special version of FORTRAN). A student who has completed Card, Printer, and Tape or Disk Programming will have little trouble working with any other hardware devices.

4. How may the new material be presented in a standard three hour per week lecture format?

Hopefully the solution to this question is represented in the course outlines included in this report. These outlines represent the lecture pattern we would follow in our first attempt to teach the material. Naturally, they are tentative outlines and would be subject to revision each semester and probably during the semester. However, we do feel themmaterial is technically and educationally correct in the order it is presented.

It should be noted that only the new courses RPG, PL/1, and ALC are outlined. The remaining courses in the suggested curriculum are or should be in present programs.

The outlines assume a block of two hours of formal laboratory time per week which should be reserved for each class, with additional time throughout the week supplied to the students. During the scheduled lab time the students will be operating the console under direction of an instructor and/or experienced lab assistant. During this time some formal instruction should be given on proper operation of the system and responses to supervisory messages.

During the remaining time, student programs should be handled on a batched basis provided the turn-around time is reasonable, or through remote terminals in a multiprogramming environment.

Two complete sets of operating manuals should be available: one at the console and one in the preparation room for student analysis of system diagnostics.

5. What effect will the third generation computer system have on attitudes to hands-on training?

With the third generation computer have come other concepts and capabilities such as multiprogramming environments, partitioning, remote terminals, etc. Many people feel that having students operating hands-on is something akin to heresy. This would perhaps be true in a commercial enterprise, but the junior college is first a learning institution. Examination of manuals and questions put to systems engineers have indicated there is no sequence of button-pushing or technique that would cause serious damage to the computer. There are many reasons supporting hands-on training in an educational institution which we feel override the objections:

1. Hands on training develops confidence and a psychological sense of mastery of the equipment.

2. Operation of the equipment helps the student to understand the system more thoroughly. Being

present when the operator messages are displayed will help the student improve his ability to function under system control.

3. Hands-on training provides immediate reinforcement of correct behavior; discouragement of incorrect behavior--the student gets immediate results of his programming and operating efforts. Time lags between performance and evaluation in educational situations are crucial.

4. Mistakes that students will undoubtedly make in machine handling are better made while at school in a training situation than after graduation when on the job and when the mistakes can be very costly to the employer. Furthermore, such mistakes, made after graduation, reflect poorly on the educational institution.

5. Hands-on operating experience is the best training in terms of preparing students to handle themselves in a professional manner when working in an operating installation.

6. The time lag between execution and results is eliminated and this facilitates more effective learning.

7. The conceptual understanding of the student is broader since it is his responsibility to diagnose his problems to completion.

SUGGESTED THIRD GENERATION COMPUTER CURRICULUM

Below is a schematic representation of the computer related courses for a two year junior college program. It does not include related courses such as mathematics, accounting, English, etc.

1st Semester	Introduction to Data Processing	
2nd Semester	Report Program Generator	Unit Record* FORTRAN IV*
3rd Semester	Procedural Languages COBOL & PL/I	Data Processing Systems and Procedures
4th Semester	Assembly Language Coding	Supervised Work Experience or Data Processing Projects
Post Certificate	Advanced System Operations	*Elective Courses

The curricula is not essentially different in structure and hours required from the curricula of second generation computer systems. There are only two modifications which tend to indicate a departure from normal procedures. The first is the reversal of instruction from the lower level machine language to the higher level languages. With the third generation computer this path has become educationally unsound. Machine language as known in the second generation is too difficult for beginning students. The second modification is the inclusion of a post certificate Advanced Systems-Operations course.

REPORT PROGRAM GENERATOR

After the introductory course the instruction will proceed from the problem oriented language Report Program Generator through the procedure oriented languages of COBOL and PL/I to the machine oriented Assembly Language Coding. The reversal from normal procedure was chosen for three reasons.

The authors believe there is a heuristic path from the easily learned Report Program Generator to the powerful and complex Assembly Language. In the beginning the instruction would be free of the tedious detail requiring both knowledge of the hardware and an understanding of system operations. Report Program Generator will enable the student to program significant problems early in his training. File organization and the use of tape and disk files in student programs may be introduced without the level of sophistication required for similar programs written in Assembly Language.

Report Program Generator is the only higher level language compatible for all IBM S/360 computers. At this writing over 25% of all computer orders are for the IBM S/360 Mod 20 Computer. RPG is the basic language of the Mod 20. At the end of the first year of instruction following the curriculum outlined, the student will be employable in many firms. Employment at this time is not an objective of the curriculum, but past experience has indicated many students find employment at this time and that they may or may not complete their education later. We feel that the working with files and RPG will help the students who find employment at the end of the first year. In addition, it will enable the returning students to be productive when they start their work experience course.

The third reason for introducing RPG first and ALC last deals with instructional and equipment considerations. Since junior college programs have more students in the introductory courses, the demand for instructors will be the largest in this area. Inservice training of instructors should be facilitated by our choice, either by the use of team teaching or the program instruction course offered by the manufacturer. The emphasis on the problem and the fixed flow chart format of RPG will make it a course easier for both student and instructor.

From an operational standpoint, RPG programs will in most cases require fewer cards in the source programs and less time to compile than most of the other languages available. The movement of student programs from conception through keypunching, compiling, debugging and execution should be facilitated. This we feel is important to the beginning student.

PROCEDURAL LANGUAGES

Emphasis, the first semester of the second year, would be placed on the procedural oriented languages of COBOL and PL/I. The effect of the third generation computers may be felt in this area. There have been many improvements in the COBOL compilers of the third generation with the merging of the compiler and the operating system. PL/I is a unique product of this generation. Both COBOL and PL/I are represented by their supportors as "The Language" of the future. This may be true; however, vocational demand for this type of training is not high at this time in our immediate placement area.

At this date there are some problems involved in teaching the procedural languages. The languages now available are given different design level ratings. Many language features are available only on certain levels. The hardware requirements of the levels varies from language to language. This means many of the options of these languages will be given a theoretical discussion, or problems must be assembled on some other machine than the one available to most junior colleges. This problem is a function of time and should decrease as more software is written. As the problem is removed for the training institution it is probable more demand for the procedural languages in the data processing community will develop.

We feel the time of the semester should be equally divided between COBOL and PL/I. This is not enough time to completely master either language, but should adequately cover the techniques of each. Many concepts of COBOL will be reinforced in the study of PL/I. The procedural language course should serve for more than just instruction in these languages. This will be the first opportunity for the students to develop their own procedures and flow charts. Two additional sets of job control cards may be covered as well as a review of files and file organization.

ASSEMBLY LANGUAGE CODING

A student entering the Assembly Language Course should be able to code almost any application he is capable of understanding. This course is designed to enable him to program the computer and provide him with an understanding of the limitations of the languages he has learned. The course is heavily hardware oriented and will be demanding on the student. The concepts of data processing and file organization should be instilled in the student prior to this class. Emphasis will be placed on the computer and the languages instead of applications.

Instruction will primarily deal with the commercial instruction set and the logical IOCS for tape and disk files. Program logic instructions will be covered, but the binary arithmetic features will be given only a cursory examination. This does not follow the presentation given at the manufacturers school, but we feel it is more consistent with actual use of the computer in immediate applications. From a learning standpoint, the commercial instruction set is easier to master and will facilitate the teaching of the functions of the Assembler Language Processor.

Mastery, both psychological and actual, of any computer comes from a knowledge of hardware and the function of each instruction. A goal of this class would bring the student to the point at which he is capable of supplying the higher level languages with specialized routines written in ALC to meet the needs of unique applications. Achievement of this goal may be difficult with the average student in one semester of machine oriented programming, but nevertheless it represents a crowning achievement.

ADVANCED SYSTEM OPERATIONS

The unique feature of this curriculum is perhaps the placement of the Advanced System Operations class. This course is placed in a post certificate position for two reasons. The material covered in the course requires the knowledge of Assembly Language Coding as a prerequisite. To move the course into the two year program would require the introductory course to be devoted exclusively to programming or the student to take two required programming courses the same semester. The former would provide a hardship on the people who take the introductory course for an overview of data processing and the latter would place additional pressures on the students.

The addition of three more units into the requirements for graduation would make the two year program in reality a two and one-half year program or would force the removal of a course such as mathematics, accounting, etc from the other half of the program. We feel the student completing the program as shown will have covered twice the data processing concepts required for graduation three years ago, and will be ready to be employed.

The second reason for an advanced course we feel will come from the community as a demand of professional programmers seeking to improve themselves. Formal training in this area is available to very few and would be required for systems programmers. The course would be taught as part of the evening division without a fixed outline. Approximately half of the course could be devoted to individual projects. Material, we feel, that should be required would include the following:

1. Generation of the system from scratch.
2. Updating and maintenance of the system.
3. The use of system utility programs.
4. Writing and testing of library routines.

It would be imperative that these students be granted hands-on time throughout the course and be given access to storage devices such as tape and disk packs.

The limitation of enrollment in Advanced Systems Operations to professional programmers will not only assure a high level course, it may also provide the instructor with an opportunity to measure the needs of the data processing community unavailable through other channels, and may also instill within the beginning student the concept that training is a continuous process and does not terminate with graduation.

ELECTIVE COURSES

There are two 9 week courses, Unit Record Equipment and FORTRAN which would be optional though suggested as electives in the program. The effect of the third generation on this training is an unanswered question. Many unit record shops are moving up to computers like the IBM Mod 20 but the electro-mechanical equipment and employment opportunities may be with us long into the future. Though many data processing concepts may be taught and crystallized with the unit record instruction, we feel the priority of this class should be reduced in a curriculum based on the third generation systems.

FORTRAN is probably the first and only universal language of computerdom. A knowledge of FORTRAN is required for many four year degree programs and may be used to meet a language requirement for some graduate degrees. However, FORTRAN has never been strong in the field of commercial data processing, being used primarily as a tool in scientific endeavors. The third generation has made little change in the language of FORTRAN or in its applications. The authors feel this course should be organized with respect to meeting the requirements of students transferring to four year schools and to serve only as general education to vocational majors.

While these classes would be considered as electives in the program, many of the concepts and terms of each would be covered in the regular classes. Report Program Generator covers master-detail card relations, sequence checking, primary-secondary files, matching records, etc. in almost the same fashion as it is handled with unit record equipment. Every concept of FORTRAN except dynamic formats are now included in PL/I. Any student completing PL/I as outlined should be able to learn FORTRAN through self study.

INTRODUCTION TO DATA PROCESSING

The function of the introductory course is to serve the needs of two types of students: the data processing major and the student who is curious about computers and data processing, and their applications. The course should be designed to present a general overview of data processing systems and their applications. Included in the course would be discussions related to the evolution, general terminology, techniques, and basic methods of data processing.

The purpose of the course is to build a general background and to orient the student to the many facets of the field. The impact of the third generation on this class is primarily in new terms and applications. Programming concepts would be introduced through the use of a machine independent simulator such as CYBERNAC.

DATA PROCESSING SYSTEMS AND PROCEDURES

The systems course is the key course in the Man Machine Management aspects of the curriculum. It is a critical course which may help the student the most after he achieves his initial entry into the vocational field. The objectives of the course are fourfold:

1. To provide detailed understanding of the interrelationships and interdependencies of dynamic systems which comprise a whole organization and the macro systems of the outside environment with which the student must integrate.
2. To provide understanding of the management processes, tools and techniques used in achieving organizational system objectives, and when and how data processing techniques can serve management most efficiently in the pursuit of these objectives.
3. To provide understanding and practice in the use or techniques of business system research, and management problem solving techniques with emphasis on the employment of the latest data processing equipment when feasible.

4. To provide understanding of the human being as the critical pivot point in the interrelationships of present organizational systems as well as in the establishment of new systems using the latest data processing equipment and techniques.

The systems class is largely hardware independent and, therefore, should be able to advance to meet the problems of a generation change. The course should now be reflecting the third generation implementational problems and devoting time to the consideration of the impact of the coming fourth generation.

The large organizational changes evolving with the advent of the fourth generation computer systems will affect the graduates of the data processing program. Vast areas of middle management and personnel will be relocated. Systems graduates will be in the midst of this relocation. In many cases, they will be asked to assume a position of leadership. This task is capable of tapping the very depths of management and technical potential of the graduating systems student.

WORK EXPERIENCE

There is clearly widespread enthusiasm for a data processing program at the community college level. Each year more companies join the ranks of those who give careful consideration to employing graduates of the occupational program in data processing. Many companies send personnel already employed in the field of data processing to the community college in the evening for advanced training. Very important to the success of the occupational program of any sort, including data processing, are opportunities for students to obtain work experience in actual installations while attending college. Companies are willing to participate in programs of this type by employing students on a part time basis. With many of these companies converting to third generation data processing systems it is desirable that educational programs emphasize those concepts which will be of the greatest value to the student and the community.

DATA PROCESSING PROJECTS

This course is designed to provide students with opportunities to gain experience in the development and implementation of scientific or commercial data processing problems. Making use of equipment available to the student, several projects are completed in which principles and techniques from previous data processing and related courses are integrated. The projects would be oriented but not limited to such areas of business activity as order writing, billing, and sales analysis, accounts receivable, purchases and cash disbursements, inventory control, payroll and labor distribution, production control, and personnel accounting.

In many cases, programming classes have a tendency to obscure the data processing problems actually encountered in the community. Laboratory projects in programming classes may be concerned with only one step in the processing of data.

The data processing projects class assists the student in focusing on the entire requirements in solving business data processing requirements. Students are required to design appropriate systems to perform several of the projects mentioned. Students would also implement their proposed systems where appropriate.

ALTERNATIVE CURRICULA

Several alternative curricula were explored prior to the formulation of the one presented. Among those given serious consideration were the following two.

1. Creation of a series of half semester nine week courses which could be taken after the completion of a basic core course. After the core course, the remaining courses could be taken in any order and not all these courses would be required for graduation. This plan is similar to manufacturer's schools. With this program the student would be provided with some freedom in shaping his education to meet his individual needs and interests.

The core course lasting a full semester would deal with an introduction to System/360 concepts and Basic Assembly Language programming. The nine week courses would include the following:

RPG
COBOL
FORTRAN
PL/I
Intermediate ALC
Support Programming Systems
File Organization Concepts
Input/Output Control Systems

It was felt there may be a staffing advantage to this program in that an instructor could concentrate in a specific area. However, there appeared to be at least three disadvantages. The requirements of the core course were judged to be too difficult for beginning students. Courses carrying 1.5 hours of credit may present problems for the transfer student. Without structure in the program, many topics would have to be taught as first time concepts rather than review items.

2. A second alternative was the reversal of the third and fourth semester programming courses, Assembly Language Coding and Procedural Languages. By this reversal the opportunity would be provided to introduce Assembly Language Coding as the expansion of user subroutines of Report Program Generator. At the end of the RPG course transition to ALC could be facilitated by pointing out the limitations of RPG and showing ALC as a possible means of alleviating the problem.

The ALC course would follow the outline as presented in this report except the first programs would be written as subroutines for RPG. With the introduction of IOCS the course would then become completely ALC. Though this natural heuristic joining of the languages is possible, it was felt the students may not at this point in their development be as ready for ALC as they would be after a semester of programming the procedural languages.

Under the suggested course outline for RPG, the last three weeks of lecture are devoted to an introduction to ALC and the hardware concepts of the 360 computer. The purpose of this is to provide a background for the student in basic hardware concepts. This background will be enriched in the procedural languages course and be fully developed in the ALC course. It is felt that early introduction of these basic concepts will help the student in his development and perhaps aid him in any independent study he may wish to do. (e.g. By using an option of PL/I it is possible to obtain an ALC listing of the program.)

UNSOLVED PROBLEMS

Between completion of formal instruction on System/360 and the development of the course outlines and curriculum, several problems as yet unsolved were observed. It was felt that these should be included in this report because they may represent the troubles of the future. Comments which follow serve only to restate the problem and perhaps suggest a solution.

1. Once a curriculum is chosen, how do we keep it from losing its perspective of the objectives of data processing and migrating to the qualitites of languages and characteristics of hardware?

This problem was prevalent in the second generation equipment instruction. It will increase geometrically in proportion to the greater complexity of third generation hardware and programming systems and to the organizational system restructuring that will be accomplished.

A thorough knowledge of languages and hardware is imperative if the student is to become productive in his initial position. However, an understanding of how advanced data processing techniques are successfully integrated into the purposeful operations of the organization is equally critical to the student's success on the job.

Because the student is heavily trained in a technical area of personnel shortages, he is often offered positions of higher organizational responsibility than the designers of curricula envisaged. Without benefit of extensive on the job coaching and guidance in integrating the equipment towards a restructuring of the organization's systems, the technically over-balanced student is capable not only of unproductivity but also of inadvertently guiding the tremendous power of data processing equipment in the opposite direction of objectives and efficiency.

Although data processing programs usually have an advance course set aside for this topic, this is not in itself a whole solution to the problem. The underlying principle of each class must involve reinforcement of the concept of integration of hardware, language, organization, and management systems. The techniques and degree of such reinforcement must be tailored into the curriculum to coincide in the intelligent support of the advanced systems courses.

2. What will be the central concepts in the data processing curriculum which will survive the transition to the next generation system?

The problem of transition to a new generation is not going to stop with the acquisition of third generation equipment and the implementation of its instruction. The fourth generation system is already being discussed in the literature and it has been estimated that it will be with us as early as 1972. The first major problem facing students after graduation may be the need for retraining for new technology. It is therefore imperative we provide in the vocational program the necessary central body of knowledge which will facilitate this retraining.

What this body of knowledge will be is difficult to ascertain. Extrapolation from the core concepts of second and third generation computer systems would indicate that the following topics should be emphasized:

1. The existence, function, similarities, differences and applications of the different type of programming systems.

Machine Oriented
Procedure Oriented
Problem Oriented
Support

2. The functions of a supervisor or monitor.

3. Input/Output control systems.

4. File organization techniques

5. Subroutines and subroutine linkage.

There will undoubtedly be changes in the terminology between the third and fourth generations as there were between the second and the present, but several of these concepts should be the same. The authors noted many questions and student difficulties at manufacturer's school appeared to concentrate about the lack of understanding of the above mentioned concepts.

3. Without a third generation computer or access to one, what can be done with the present curriculum that would most aid the student who must make the transition to the third generation after graduation?

For approximately three weeks during the Spring of this year the above problem was the major concern of the authors. Several conferences were held dealing with what could be done if there were no access to a third generation computer. During this time many avenues were explored. No ultimate solutions were obtained and the search was abandoned when the problem was ameliorated through district acquisition of System/360 time from a local firm. Our efforts were then directed to the formulation of the curriculum shown in the last section of this report.

However, some of the results of our search merit consideration and may prove useful to other data processing departments who face similar circumstances.

We concluded that it would be difficult to effectively teach the hardware oriented concepts and many of the system concepts of the third generation computer without actual laboratory experience. With experienced programmers a theoretical discussion of hardware and systems would perhaps be sufficient, but with beginning students who were also learning programming techniques, it was felt that the actual mastery of an available computer would be of greater practical benefit.

At first the possibility of writing a System/360 simulator for the IBM 1620 was considered. The simulator would have been similar to the 141 simulator of 1401 fame, written by the late Kenneth P. Swallow. This project was shelved before it was completely explored. It did appear that perhaps simulating the entire instruction set would have been too ambitious, but the Commercial Instruction Set and certain other instructions could have possibly been done. It should be noted that certain programs linking the System/360 to second generation machines are now appearing, (e.g. File 1401.1.056 Edit System/360 Basic Assembly Language on 1401/1440). At the time of this writing the authors have not had the opportunity to examine any of these programs.

The program envisaged would have been similar to the present Orange Coast College curriculum except the last semester would concentrate its programming efforts on the IBM 1620 computer with theoretical disucssions given on Assembly Language of the 360. The functions of the Monitor would have been examined and attempts to show parallels with System/360 would have the major priority. The program would not have been as effective as the present Orange Coast College program should be, but it would have represented a move forward to meet the problem.

The IBM 1620 computer which is available to many junior colleges may be utilized to teach several concepts which are found in System/360. With a Disk and the Monitor of the IBM 1620 as a basis, a programming course, or courses could be developed which would provide a background that would enable the student to make the transition to System/360 without having to master completely new material. The student would learn computer programming on an available computer and at the same time receive a foundation which would assist him later.

Hardware similarities between the two computers which lead to parallel programming techniques are the following:

Boundary alignment conditions.
Character and decimal formats.
Condition settings reflecting arithmetic operations.
Movement from right to left operands.
Immediate instructions.

With a little effort software routines could be added to the SPS library of the 1620 which would simulate the instructions of the 360 such as EDIT, BXLE, BCT and others.

From a system standpoint, while the names are not the same, there are many similar functions and concepts between the Monitor of the 1620 and System/360. The following is a partial list of items available to both systems:

A basic monitor or supervisor.
A group of language translators.
An IOCS get/put system.
A library system.
A linkage edit routine.
Job control language.
Utility routines.

Operator intervention
System messages
Storage of information relative to use
Job stream input

However, there are several concepts and functions not abailable for demonstration by use of the 1620. Included among these are:

Interrupts
Assignment cards
Data sets
File organization support
Complete IOCS (Physical & Logical)
Multiprogramming
Time sharing
General purpose registers
Base displacement addressing

It should be noted that the IBM 1620 is a second generation computer and it is not equivalent to the IBM System/360. Data processing instruction based on the 1620 alone would not fulfill the requirement that the student be capable of productive work in a System/360 programming position. Training utilizing the 1620 with appropriate references to System/360 would benefit the student, but he would not in the opinion of the authors be at the same level of sophistication or productivity as a student having the opportunity to program actual third generation equipment.

4. How will the junior college meet the initial and continual learning requirements of data processing staff members:

A junior college, even though it has both a third generation computer and a program for instruction, cannot meet the needs of its students and community without an adequately trained staff. The staff training of data processing department personnel can be broken down into two catagories: initial training, and on the job training. Each phase of this training merits some consideration in this report.

The initial training of data processing instructors has in the past been quite varied. Due to the newness of the field, few if any formal colleges or universities have developed teacher training programs. Instructors presently in the field have primarily received their training through:

1. Industrial work experience
2. Manufacturers' schools
3. Programmed instruction courses
4. Inservice or summer institutes
5. Team teaching
6. Independent study

Third generation computers with their accompanying systems and applications require retraining of most existing data processing instructors and extended training for new personnel. The initial training period is longer due to the increased body of knowledge and applications. Since the field is dynamic and growing at a fantastic rate, with the increased teacher demand, mastery of the subject at the time an individual starts teaching is in many cases impractical.

Continuing on the job training is imperative for all data processing instructors. The instructors must not only be aware of new developments in language and applications, they must also be knowledgeable enough in these areas to make judgments as to how, where, and when these new concepts are to be integrated into the educational program. A theoretical knowledge of the equipment and techniques is not enough in a hands on training situation. The instructor must master idiosyncrasies of the equipment and language for proper operational procedures and debugging those diagnostics and problems the manuals fail to cover.

As the data processing field expands, text books lag. The instructor is forced to provide supplementary material and examples for his lectures. New texts on the third generation are starting to appear, but many are restatements of the manuals with few examples. At present the major sources of programming techniques and applications are professional periodicals and meetings.

There are five basic ingredients for an adequate staff. The first three deal with the individual instructor and are prerequisites for success in any field: Apptitude, motivation, and willingness to work. The remaining two are free time and opportunity for training. The first three are found in most data processing instructors. The last two are provided by the school and the community.

The continual learning and updating requirements faced by the data processing staff are not similar to those made by the mathematics staff a few years ago. Both authors made the transition from the "Traditional" to the "Modern" mathematics. This change was in many cases a change in terms only. The content and applications of the majority of junior college mathematics class remained the same. The problem faced in data processing is one of continual change. With each technological and management advance, new applications are developed requiring a new body of knowledge. The data processing department which fails to reflect the advances in the commercial world is soon shortchanging its students and not meeting the objectives of vocational training.

In addition to learning and remaining current in the field of data processing, the instructor is faced with a unique problem in the area of student assistance. Programming classes in particular require substantial time. The initial training usually gives the instructor a basic understanding of the language. This knowledge may be sufficient for the coding of a lab exercise but it is not sufficient to answer student inquiries or assist students debug techniques which appear correct, but fail to work.

The instructor must not only have the skill to develop the logic for solving a given problem, he must also be able to follow the logic of students. Instructor or student misinterpretation of a manual passage can sometimes take hours to straighten out. Analysis of core dumps is often the court of last resort.

Since a close relationship between the instructor and the students is desirable at the junior college level, the junior college is faced with the problem of providing a sufficient amount of free time for the instructor to become an "expert" in his area.

5. What new computer applications may have major implications in the future for the data processing staff and the junior college.

New applications are appearing with great rapidity in the field of computers and data processing. Continual monitoring of announcements and evaluation of new innovations are essential if the vocational data processing program is to keep its forward movement with technology. Three new innovations, Computer Assisted Instruction, Interpretive Languages and APL\360, are not now considered in the realm of vocational data processing; however, they may be the major topics in the future.

Computer Assisted Instruction is being evaluated as an educational tool by many colleges. The concept of using the computer to monitor the progress of a student and tailor the course material to the individual student is indeed interesting. At present there are some courses conducted entirely through the use of remote terminals and a computer. Two schools, University of California, Irvine, and University of Oklahoma have several classes conducted under CAI. The Medical School of University of Oklahoma uses a CAI program to handle the enrollment of its students. With System/360, CAI may become important to the junior college both as an educational tool and for its administrative applications. CAI should merit consideration in the vocational data processing program.

The basic language of CAI, COURSEWRITER, is easy to learn and could adequately be covered in two weeks of lecture. The writing of an effective course of instruction to be given under CAI is difficult. The writer must be skilled in the field of instruction and in techniques of education. Courses given under CAI should be written by qualified instructors with the assistance of programmers. CAI may be the catalyst which will involve the entire staff of a junior college in the data processing program.

The interpretive languages QUICKTRAN, DATATEXT, and A Programming Language are systems which utilize a large central computer and several remote terminals. Of the three, DATATEXT is probably the most vocationally oriented.

It links an office typewriter to a remote computer thereby enabling a secretary to utilize the computer to handle repetitious typing and editing of documents. The actual instruction of DATATEXT may be under Secretarial Science, but the data processing staff should continually evaluate its potential.

QUICKTRAN, a FORTRAN like language, is currently used by many schools. With QUICKTRAN a student may utilize a remote terminal as a large scale desk calculator. Vocational demand for QUICKTRAN is not high in our area, but some Orange Coast College students have found employment in shops using General Electric's interpretive language, BASIC.

A Programming Language, APL\360, developed by Dr. K. E. Iverson of IBM, may have great educational value in the area of mathematics. APL\360 is a new development, as yet not released, which may or may not affect the vocational curriculum. Like the other new applications, it must be evaluated. The authors see great potential in APL\360 mathematics instruction. The vocational applications of APL\360 will probably lie in the areas of Operations Research and Linear Programming. At the present time neither of these areas are covered in the junior college.

THE DATA PROCESSING PROGRAM AT ORANGE COAST COLLEGE 1967-68

The Orange Coast Junior College District ordered an IBM System/360 Model 40 in 1965 for expanding administrative and educational requirements. The scheduled delivery date was for September, 1967. The authors went to the various IBM schools during the school year 1966-67 in anticipation of delivery of System/360.

Due to financial considerations, the college district has had to postpone delivery of System/360. The district currently has an IBM 1401 Disk System which is shared by the administrative and educational elements of the school district. In addition, the school district owns an IBM 1620 Disk system which is employed exclusively by the educational data processing department.

In 1966-67, there were approximately 500 day and evening students enrolled in data processing courses each semester. The data processing courses were oriented toward second generation concepts. In the programming courses, emphasis was placed on 1401 machine oriented languages.

The data processing courses in the curriculum last year are schematically presented below:

1st Semester	Introduction To Data Processing	
2nd Semester	CYBERNAC 1620 Programming Machine Language, SPS, FORTRAN II 1401 Machine Language	Unit Record
3rd Semester	1401 SPS 1401 Autocoder 1401 DISK I/O	Data Processing Systems
4th Semester	1401 MACROS 1401 IOCS COBOL FORTRAN IV	Data Processing Projects* and Work Experience*

*Electives

It became apparent last year that significant changes in the data processing program were in order. The college district is vitally concerned with providing meaningful vocational programs. The time to move forward was now!

In discussing the possible changes in the program, it was decided that:

1. The philosophy of hands on training operating within an open shop should be continued, particularly in the first programming classes.
2. The changes should not have a disturbing effect on the students currently in the program.

The district had a distinct advantage in that the instructors were enthusiastic and highly motivated to make changes in the program. Several of the instructors were on the staff when the second generation concepts were introduced and expanded. It was also recognized that the changes were necessary in order to be prepared for the inevitable transition to 4th generation computers in the future.

One of the major decisions reached was to drop all 1401 machine oriented languages (i.e. machine language, SPS, Autocoder, Macro programming, and IOCS) from the programming courses. Programming languages such as RPG, PL/I and System/360 ALC would be integrated into the programming classes to replace 1401 machine oriented programming.

The data processing courses at Orange Coast College at the present time are presented as follows:

1st Semester	Introduction To Data Processing	
2nd Semester	1620 Programming RPG	Unit Record
3rd Semester	Procedural Languages: FORTRAN IV COBOL PL/I	Data Processing Systems
4th Semester	System/360 ALC	Data Processing Projects* and Work Experience*

*Electives

The programming courses represent a partial implementation of the suggested curriculum. A complete transition in the programming class has not been made due to 1620 availability for hands on training and its benefit to beginning students.

Since there are many similarities between the 1620 and System/360, it was decided that the first programming class should include machine oriented (machine language and SPS), FORTRAN II, and operating system principles employing the 1620 system. This would allow beginning students to gain hands on experience and the satisfaction of executing and debugging their own programs. Approximately one-third to one-half of the course is devoted to a discussion of RPG. Initial plans call for students to write programs in RPG in which the instructor will review these programs, make corrections, and offer suggestions.

The second programming class involves the procedural oriented languages. FORTRAN IV and COBOL programs are executed on the 1401 system under an open shop arrangement. Due to the length of compile time (minimum time 20 minutes) in COBOL, FORTRAN IV is also included since many of the principles in PL/I are found in FORTRAN IV. PL/I will occupy approximately one-third of the course and will be conducted similarly to RPG.

The college has made arrangements with a local firm to utilize their System/360 for processing programs in the third programming class. Programs written in 360 ALC will be delivered to this firm three times a week. Turn around time will be a maximum of two days. The normal time is expected to be one day. In this manner, students will be reinforced in the learning process by actually running programs written for System/360. There is no hands on instruction involved in this class, however, students are at this point in time psychologically prepared for closed shop programming in ALC.

After postponing System/360, district funds were budgeted for the processing of ALC programs. This step was taken in recognition of the new technology and the desire to offer the best possible program within the financial resources of the district.

The contents of other data processing courses are being revised to reflect the viewpoints expressed in this report.

In addition to the data processing courses outlined, the following related courses are also required for the Associate of Arts Degree:

Accounting 1
Accounting 2
Accounting 61 (Cost Accounting)
Math 33 (Data Processing Math)

Data processing curriculums require frequent revision and modifications. Additional modifications will be made to the data processing program at Orange Coast College. The changes will reflect the judgement of those who guide the program and will be made to the end that the program will serve the best interests of the students and the community.

SUGGESTED COURSE OUTLINES
FOR
REPORT PROGRAM GENERATOR
PROGRAMMING LANGUAGE I
ASSEMBLY LANGUAGE CODING

Suggested course outline Report Program Generator

Week 1 Introduction

- Terminology of System/360 data representation
  - Bits & Bytes
  - EDCDIC Characters
  - Binary & Hex representation

- Numeric representation
  - Zoned decimal format
  - Packed decimal format
- Need for a language to work with data

- Basic elements of RPG programming
- Functions of RPG
- Generalized RPG flowchart

Week 2 RPG Card to Printer Programming

- Program coding sheets overview
- File Description Specifications
- Input Description Specifications
- Output Description Specifications
- Write 80-80 list program

- Input Description Specifications
  - Card field descriptions
  - Stacker select
  - Character input
  - Packed input

- Output Description Specifications
  - Heading lines
  - Detail lines
  - Page overflow

Report Program Generator

Week 3 RPG Card to Printer Programming

- Input Description Specifications
  - Master-Detail cards
  - Indicator codes
    - Record/identifier Codes
    - Character/decimal/zone formats
  - Field indicators
    - Plus/minus/zero/blank

- Editing Output
  - Literals
  - Edit words
    - Body entries
    - Status entries

- Calculation Specifications
  - Decimal alignment
  - Storage definition
  - Zero and add Z-ADD
  - Addition ADD
  - Subtraction SUB

Week 4 RPG Programming General Computations

- Calculation Specifications
  - Multiplication MULT
  - Division DIV
  - Half adjustment

- Logic Operations
  - Compare Numeric data
  - Compare alphanumeric data
  - Field indicators
    - High/low/equal

- Examination I

Report Program Generator

Week 5 RPG Programming Multiple Files

- Review examination
- Move commands
    - Move MOVE
    - Move Remainder MVR
    - Move Left MOVEL
    - Move Zone Commands

- Primary/Secondary files
- Order of reading records
- Matching records

- Indicator codes
- Using indicator codes in calculation specs
- Input records in an OR relation

Week 6 RPG Programming Multiple files output

- Levels of totals
    - Minor/Intermediate/Major
    - Final total

- File control
    - Page numbering
    - Printing lines conditioned by overflow

- Methods of Printing
    - Detail
    - Group
    - Group Indication
    - Footnotes

Report Program Generator

Week 7 Tape Storage Concepts System/360

Physical characteristics
System Assignments
Labels
Label checking

Opening tape files
Closing tape files
Data formats
Blocking

Examination II

Week 8 RPG Programming Tape Files

Review examination
File Description Specifications
Input Description Specifications
Tape records

Retrival of records from master tape
Creation of new updated master tape
Retrival of selected records from master tape

Testing RPG program switches
Setting RPG program switches
Branching techniques in RPG

Report Program Generator

Week 9 Tables

- Tables in general
- File Extension Sheets
- Table names
- Rules for forming tables

- Table operations
- Table retriving
- Table updating

- Methods of processing tables
  - Card RPG systems
  - Disk/tape RPG systems

Week 10 Disk Storage Concepts

- Physical organization
- 2311 Storage concepts
- Count/Data records
- Count/Key/Data records

- Sequential file organizations
- Direct file organizations

- File Description Specifications
- File Extension Specifications

Week 11 RPG Programming Disk Files

- Programming sequential files
- Programming direct files
  - Sequential processing
  - Random processing

- Examination III

- Review Examination
- Introduction to Index Sequential Files

Report Program Generator

Week 12 RPG Programming Index Sequential Files

- Index Sequential Files
  - Physical concepts

- Processing Techniques Index Sequential Files
  - Sequential processing
    - Entire file
    - Within limits
  - Random processing
  - Chaining files

- Updating files at detail time
- Updataing files at total time
- Chaining files at detail time
- Chaining files at total time

Week 13 Multiple File Processing

- Sequential processing multiple files
- Matching record techniques

- Random processing multiple files
- Chaining techniques

- Examination IV

Week 14 Introduction System/360 Assembly Language

- Review examination
- 360 core representations

- General purpose registers
- Addressing scheme
- Data representations

- Functions of machine oriented language
- Relation of ALC to RPG
- Responsibilities of programmers under ALC

Report Program Generator

Week 15 Assembly Language Coding

Assembly Language coding sheet
   Labels
   Operands
   Operators
Declaratives for packed decimal

Arithmetic commands	core to core
Zero and add packed	ZAP
Add-decimal	AP
Subtract decimal	SP

Condition codes

Branch conditional
Unique mneumonics

Arithmetic commands	core to core
Multiply packed	MP
Divide packed	DP

Week 16 Subroutines

Move numeric
Move with offset
   Truncation techniques
   Rounding techniques

Base displacement notation
Subroutine linkage RPG to ALC
Techniques for loading registers
Techniques for saving data

Program locic manual of RPG
   Examination of coding generated by RPG

Week 17 & 18

Reserved to provide
time to slip on the schedule
and for holidays.

Suggested Course Outline For PL/I Programming

Week 1

- General philosophy and functions of PL/I
- Concepts of defaults
- 48 character vs 64 character set
- Punctuation
- Comments
- Identifiers

- Variables
    - Binary
    - Character
    - Fixed
    - Decimal
    - Logical
    - Floating
- Declare statements & defaults

Base	Decimal	or Binary
Scale	Fixed	or Floating
Mode	Real	or Complex
Precision	(w,d)	or (w)

- Declare statements
- Table entries
- Subscripts
- Initialization
- Order of efficiency of arithmetic

Week 2

- Assignment statements
- Operations
    - Arithmetic
    - Comparison
    - Bit String
- Prefix operators
- Infix operators

- Labels
- Mixing mode of operands
- Conversion problems

- Comparison operators
    - Arithmetic
    - Character
- Logical operators
    - And - Or
    - Switch testing
- Concatenation
- Compiling messages from table entries

PL/I Programming

Week 3

- Control statements
  - GO TO
  - STOP
  - EXIT
  - Simple IF
  - Alternate IF
  - Nested IF

- Input/Output statements
- Overview of Stream and Record I/O
- Stream oriented pipeline concepts
- List directed I/O characteristics
  - GET LIST options
  - PUT LIST options
  - Horizontal & vertical output spacing

- Data directed I/O characteristics
  - GET DATA options
  - PUT DATA options
- DISPLAY --operator messages
  - REPLY
  - EVENT
  - WAIT

Week 4

- DO statements
  - DO Group
  - DO WHILE
  - DO Counter
  - Nested DO
  - DO Counter ... WHILE

- Scaler operations on arrays
- Statement label variables
- Many position switches

- Examination

PL/I Programming

Week 5

- Arrays with different attributes
- Major Structure names
- Levels & level numbers
- Minor structure names
- Elementary item names
  - Attributes
- Qualified names

- Manulipulation of data in structures
  - Serially
  - Selevtively
  - Reformatting
- LIKE attribute

- DEFINE attribute
- PICTURE attribute
- Picture specification characters
- Editing techniques
- Execution time restrictions

Week 6

- Edit directed I/O
- GET EDIT options
- Input formats
- PUT EDIT options
- Output formats
- Control format items

- Variable formats
- Remote formats
- Format list
- Techniques for dynamic formats
- Techniques for card input

- File description attributes
- File attribute
- Function attributes
- Access attributes
- Keyed attributes
- Environment attributes
- Standard defaults

PL/I Programming

Week 7

- Interrupts
  - Interrupts which always occur
  - Interrupts which never occur
  - Interrupt actions
- ON conditions
- BEGIN blocks
- Forcing conditions

- Scope of the ON statement
- Scope of Conditional prefix
- Programmer defined ON conditions

- Subroutines & Functions
- Arguments
- Scope of variable name
- Common data
- Comparison of subroutine & functions
- Techniques on bypassing attribute differences in calling program and subroutine

Week 8

- External & internal procedures
- Flow of program control
- Common data
- Programmer control over storage
- Static storage
- ALLOCATE statement
- FREE statement

- Tasking
- Serial processing
- Asynchronous processing
- EVENT option
- TASK option
- PRIORITY option
- Switching priority

Examination

Suggested Course Outline Assembly Language Coding

Week 1

- Introduction System/360
- Core representation
- Addressing scheme
- Data representation
    - Packed decimal
    - Zone decimal
- Declaratives: Packed Decimal
    - DC P' '
    - DS P
    - Length Attribute

- Arithmetic commands
    - ZERO AND ADD ZAP
    - ADD DECIMAL AP
    - SUBTRACT DECIMAL SP
- A.L.C. Coding sheets
    - Comments
    - Labels
    - Operations
    - Operands
- Technique for loading base register

- MULTIPLY DECIMAL MP
- Clearing an area to zero
- Condition Code settings after arithmetic commands
- BRANCH CONDITIONAL
- Unique branch conditional mnemonics

Week 2

- Writing with base-displacement-length notation
- MOVE WITH OFSET MVO
- MOVE NUMERIC MVN
- Truncation techniques
- Rounding techniques

- Student INPUT/OUTPUT Macros
    - RCARD
    - WLINE
    - SK
    - STOP
- Conversion to & from zoned decimal
    - PACK PACK
    - UNPACK UNPK

- DIVIDE DECIMAL DP
- Write a program similar to Lab 2

Assembly Language Coding

Week 3

- Techniques for reading a core dump
- Program Status Word
- Interrupts --review

- Editing techniques and procedures
- EDIT — ED
- Pattern field
- S-trigger & rules
- Fill Character
    - Character --Hex representation
    - DC X' '
- Condition code settings

- Character Constants
    - DC C' '
    - DS C
- MOVE CHARACTER — MVC
- Techniques for blanking an area
- Techniques for zero ballance

Week 4

- EDIT AND MARK — EDMK
- Floating & techniques
- * protection techniques
- MOVE (IMMEDIATE) — MVI
- Review general purpose registers
- Table construction

- TRANSLATE AND TEST — TRT
- Techniques for working with short tables
- Extracting fields from compacted records

EXAMINATION I

Assembly Language Coding

Week 5

Review examination	
Indirect addressing techniques	
OR logic	EX
EXECUTE	

Logic commands	
AND	N
OR	O
EXCLUSIVE OR	X
USING	
BRANCK AND LINK	BAL

Program switches	
Branch no-op switches	
Character switches	
Bit switches	
COMPARE LOGICAL (IMMEDIATE)	CVI
INSERT CHARACTER	IC
TEST UNDER MASK	TM

Week 6

LOAD	L
STORE	ST
Address constants	
Literals	
STORE MULTIPLE	STM
LOAD MULTIPLE	LM

Subroutines
Arguments
Temporary areas
Return
Multiple entries
Multiple exits

System/360 control cards
    function of cards used by students
Linkage Editor
Core Image Library

Assembly Language Coding

Week 7

- Introduction to functions of IOCS
- Physical IOCS
- Logical IOCS
- Channels

- Functions of IOCS macros
  - OPEN
  - GET
  - PUT
  - CLOSE

- Card file IOCS
  - DTFCD entries
  - IOMOD for card
- Printer file IOCS
  - DTFPR entries
  - IOMOD for printer
- Program for 80-80 list using IOCS

Week 8

- Variable length records
- Printer control macros
  - CNTRL
  - PRTOV
- Page headings

- Control Sections CSECT DSECT
- Begin Literal Pool LTORG
- Job Control cards
- Job steps

EXAMINATION II

Assembly Language Coding

Week 9

Review examination	
TRANSLATE	TR
Reformatting records	

Tape
Physical characteristics (review)
Blocking
I/O areas
Record formats
Multi reel file & Multi file reel

Tape file IOCS
    DTFMT entries
    IOMOD for tape
Functions of OPEN and CLOSE

Week 10

Tape file IOCS
    DTFMT entries continued
Tape label formats
Variable length records
    TRUNC
    VARBLD

Write program in class building a tape file from a card file
    with blocked fixed length records
    with variable length records

COMPARE BINARY	C
COMPARE LOGICAL	CL
Looping concepts	

Assembly Language Coding

Week 11'

Looping control	
BRANCH ON COUNT	BCT
BRANCH ON INDEX HIGH	BXH
BRANCH ON INDEX LOW OR EQUAL	BXLE

EXAMINATION III

Review examination
External subroutine linkage
INCLUDE

Week 12

Disk Storage Device
Physical characteristics
Addressing scheme
- CCHHR
- Count-Data
- Count-Key-Data

Disk Labels
Volume labels
Volume Table of Contents
Formats of file labels
DASD label statements

Function of OPEN & CLOSE of disk I/O
Direct access files
Randomizing techniques

Assembly Language Coding

Week 13

- Disk file IOCS
    - DTFDA entries
    - IOMOD for disk
- Building a random file
- Retriving a random file

- Updating a random file
- Additions to a random file
- Blocking techniques

- Disk file IOCS
    - DTFSD entries
    - IOMOD for disk

Week 14

- User interrupt macros
    - STXIT
    - EXIT
- CONVERT TO BINARY CVB
- CONVERT TO DECIMAL CVD

EXAMINATION IV

- Review examination
- Index Sequential File Organization
    - Physical characteristics

Assembly Language Coding

Week 15

- Disk file IOCS
    - DTFIS entries
    - IOMOD for disk
- Retrival of Index Sequential records
    - Selected records
    - All records

- Building an Index Sequential File
- Updating an Index Sequential File

- Additions to an Index Sequential File

Week 16 Open days for review, holidays
Week 17 and to provide extra time for
Week 18 more discussion of certain topics.

PROGRAMMING EXAMPLES
AND
CLASS ROOM HANDOUTS
FOR
REPORT PROGRAM GENERATOR

RPG INTRODUCTORY EXAMPLE

INPUT	FIELD NAME	CARD COLUMNS
	Customer Number	1 - 5
	Class	6 - 8
	A/R Balance	9 - 16
	Discount Rate	17 - 19
	Name	20 - 39
	Address	40 - 59
	City and State	60 - 79

OUTPUT DETAIL PRINTED REPORT

CUST. NO.	CLASS	NAME	DISCOUNT RATE	A/R BALANCE
1 - 5	9 - 11	15 - 34	40 - 42	47 - 56

Print Positions

REQUIRED: Read input cards and produce a detail printed report. Also, accumulate and print a final total on Accounts Receivable balance.

LINE DESCRIPTION FIELD HEADINGS WORD MARKS 6 Lines Per Inch IBM 407, 408, 409, 1403, 1404, 1443, nd 220

R.P.G. CLASS
INTRODUCTORY
EXAMPLE

GLUE

```
* H   4  CUST.  CLASS         NAME           DISCOUNT    A/R
  H   5   NO.                                  RATE    BALANCE

  D   8  XXXXX   XXX   XXXXXXXXXXXXXXXXXXXX    .XXX   XXX,XXX.XX
T(LR) 10                                          $XX,XXX,XXX.XX ***
```

IBM

INTERNATIONAL BUSINESS MACHINES CORPORATION

REPORT PROGRAM GENERATOR FILE DESCRIPTION SPECIFICATIONS

IBM System 360

Printed in U.S.A.

Date ______

Program RPG INTRODUCTORY EXAMPLE

Programmer C. D. GAULLE

Punching Instruction			
	Graphic	0	O
	Punch	N	A

Page 01

Program Identification INTROD

Line (3–5)	Form Type (6)	Filename (7–14)	File Type I/O/U/C (15)	File Designation P/S/C/R/T (16)	End of File E (17)	Sequence A/D (18)	File Format F/V (19)	Block Length (20–23)	Record Length (24–27)	Mode of Processing L/R (28)	Length of Record Address Field (29–30)	Record Address Type K/I (31)	Type of File Organization I/D/T (32)	Overflow Indicator (33–34)	Key Field Starting Location (35–38)	Extension Code E/L (39)	Device (40–46)	Symbolic Device (47–52)	Labels (S, N, or E) (53)	Name of Label Exit (54–59)	Extent Exit for DAM (60–65)	Comments (66–74)
010	F	CARDIN	I	P			F	80	80								READ40	SYSIPT				
020	F	PRINTOUT	O				V	132	132					OF			PRINTER	SYSLST				
03	F																					
04	F																					
05	F																					
06	F																					
07	F																					
08	F																					
09	F																					
10	F																					
11	F																					
12	F																					
13	F																					
14	F																					
15	F																					
	F																					
	F																					
	F																					
	F																					
	F																					

INTRO I

REPORT PROGRAM GENERATOR INPUT SPECIFICATIONS
IBM System 360

Date

Program

Programmer

Punching Instruction	Graphic							
	Punch							

Page 02

Program Identification INTROD

Line	Form Type	Filename	Sequence	Number (1-N)	Option (O)	Resulting Indicator	Record Identification Codes 1 Position	Not (N)	C/Z/D	Character	2 Position	Not (N)	C/Z/D	Character	3 Position	Not (N)	C/Z/D	Character	Stacker Select	Packed (P)	Field Location From	To	Decimal Positions	Field Name	Control Level (L1-L9)	Matching Fields or Chaining Fields	Field-Record Relation	Field Indicators Plus	Minus	Zero or Blank	Sterling Sign Position
010	I	CARDIN	AA			01																									
020	I																				1	5	0	CUSTNO							
030	I																				6	8	0	CLASS							
040	I																				20	39		NAME							
050	I																				17	19	3	DISCT							
060	I																				9	16	2	ARBAL							
07	I																														
08	I																														
09	I																														
10	I																														
11	I																														
12	I																														
13	I																														
14	I																														
15	I																														

INTRO 2

IBM

INTERNATIONAL BUSINESS MACHINES CORPORATION

REPORT PROGRAM GENERATOR CALCULATION SPECIFICATIONS

IBM System 360

24-3351-1
Printed in U.S.A.

Date ____________

Program ____________

Programmer ____________

Punching Instruction	Graphic							
	Punch							

Page 1 2: 03

Program Identification 75 76 77 78 79 80: INTROD

Line 3 4 5	Form Type 6	Control Level (L0-L9, LR) 7 8	Indicators: Not 9 10 11	Indicators: And Not 12 13 14	Indicators: And Not 15 16 17	Factor 1 18–27	Operation 28–32	Factor 2 33–42	Result Field 43–48	Field Length 49 50 51	Decimal Positions 52	Half Adjust (H) 53	Resulting Indicators: Plus / Compare: High 1 > 2 54 55	Resulting Indicators: Minus / Compare: Low 1 < 2 56 57	Resulting Indicators: Zero / Compare: Equal 1 = 2 58 59	Comments 60–74
010	C		01			ARBAL	ADD	FINAL	FINAL	10	2					
02	C															
03	C															
04	C															
05	C															
06	C															
07	C															
08	C															
09	C															
10	C															
11	C															
12	C															
13	C															
14	C															
15	C															
	C															
	C															
	C															
	C															
	C															

INTRO 3

IBM

INTERNATIONAL BUSINESS MACHINES CORPORATION

REPORT PROGRAM GENERATOR OUTPUT-FORMAT SPECIFICATIONS

IBM System 360

Date ______

Program ______

Programmer ______

Punching Instruction		
	Graphic	0
	Punch	A

Page 04

Program Identification INTROD

Line	Form Type	Filename	Type (H/D/T)	Stacker Select	Space Before	Space After	Skip Before	Skip After	Output Indicators Not	Output Indicators	And Not	And	And Not	And	Field Name	Zero Suppress (Z)	Blank After (B)	End Position in Output Record	Packed Field (P)	Constant or Edit Word	Sterling Sign Position
010	O	PRINTOUT	H			1	01			1P											
020	O		OR							OF											
030	O																	12		'CUST. CLASS'	
040	O																	25		'NAME'	
050	O																	52		'DISCOUNT A/R'	
060	O		H			3				1P											
070	O		OR							OF											
080	O																	4		'NO.'	
090	O																	54		'RATE BALANCE'	
100	O		D			1				01											
110	O														CUSTNO	Z		5			
120	O														NAME			34			
130	O														CLASS	Z		11			
140	O														ARBAL			56		' , , 0. '	
150	O														DISCT			42		'.	
160	O		T		1			01		LR											
170	O														FINAL			60		'$, , 0. &***'	
	O																				
	O																				
	O																				

INTRO

LINE DESCRIPTION

FIELD HEADINGS/WORD MARKS

6 Lines Per Inch

IBM 407, 408, 409, 1403, 1404, 1443, and 2203

ACCOUNTS RECEIVABLE
REGISTER
PRACTICE PROBLEM #1

GLUE

```
X H  6             ACCOUNTS     RECEIVABLE     REGISTER
     7
  H  8 CUSTOMER                            LOCATION   INVOICE     INVOICE          INVOICE
  H  9  NUMBER    CUSTOMER NAME           STATE CITY   NUMBER      DATE             AMOUNT
    10
  D 11  XXXXX     X--------------------X    XX  XXX    XXXXX      X0/XX/XX        XX,XX0.XX
    12
    13
    14
T(LR)15                                                                      $X,XXX,XX0.XX *
    16
    17   ZERO  SUPPRESS:
    18     CUST  NO
    19     STATE
    20     CITY
    21     INVC  NO
    22
    23  DOUBLE  SPACE  DETAIL  LINES.
```

IBM

INTERNATIONAL BUSINESS MACHINES CORPORATION

REPORT PROGRAM GENERATOR FILE DESCRIPTION SPECIFICATIONS

IBM System 360

Date ____________

Program PRACTICE PROBLEM #1

Programmer ____________

Punching Instruction			
	Graphic	O	Ø
	Punch	A	N

Page [illegible]

Program Identification PROB1

Line	Form Type	Filename	File Type I/O/U/C	File Designation P/S/C/R/T	End of File E	Sequence A/D	File Format F/V	Block Length	Record Length	Mode of Processing L/R	Length of Record Address Field	Record Address Type K/I	Type of File Organization I/D/T	Overflow Indicator	Key Field Starting Location	Extension Code E/L	Device	Symbolic Device	Labels S, N, or E	Name of Label [illegible]	[illegible]	[illegible]
3 4 5	6	7 8 9 10 11 12 13 14	15	16	17	18	19	20 21 22 23	24 25 26 27	28	29 30	31	32	33 34	35 36 37 38	39	40 41 42 43 44 45 46	47 48 49 50 51 52	53	54		
0 1	F	ACCREC	I	P			F	80	80								READ40	SYSIPT				
0 2	F	REPORT	O				V	132	132					OF			PRINTER	SYSLST				
0 3	F																					
0 4	F																					
0 5	F																					
0 6	F																					
0 7	F																					
0 8	F																					
0 9	F																					
1 0	F																					
1 1	F																					
1 2	F																					
1 3	F																					
1 4	F																					
1 5	F																					
	F																					
	F																					
	F																					
	F																					
	F																					

IBM

INTERNATIONAL BUSINESS MACHINES CORPORATION

REPORT PROGRAM GENERATOR INPUT SPECIFICATIONS

IBM System 360

Date

Program

Programmer

Punching Instruction								
	Graphic							
	Punch							

Page 02

Program Identification PRO01

Line	Form Type	Filename	Sequence	Number (1-N)	Option (O)	Resulting Indicator	Record Identification Codes												Stacker Select	Packed (P)	Field Location		Decimal Positions	Field Name	Control Level (L1-L9)	Matching Fields or Chaining Fields	Field-Record Relation	Field Indicators			Sterling Sign Position
							1				2				3																
							Position	Not (N)	C/Z/D	Character	Position	Not (N)	C/Z/D	Character	Position	Not (N)	C/Z/D	Character			From	To						Plus	Minus	Zero or Blank	
3 4 5	6	7 8 9 10 11 12 13 14	15 16	17	18	19 20	21 22 23 24	25	26	27	28 29 30 31	32	33	34	35 36 37 38	39	40	41	42	43	44 45 46 47	48 49 50 51	52	53 54 55 56 57 58	59 60	61 62	63 64	65 66	67 68	69 70	71 72 73 74
0 1	I	ACCREC	AA			01																									
0 2	I																				39	43	0	CUSNO							
0 3	I																				8	29		CUSNAM							
0 4	I																				44	45	0	STATE							
0 5	I																				46	48	0	CITY							
0 6	I																				34	38	0	INVNO							
0 7	I																				49	54	0	INVDAY							
0 8	I																				74	80	2	INVAMT							
0 9	I																														
1 0	I																														
1 1	I																														
1 2	I																														
1 3	I																														
1 4	I																														
1 5	I																														
	I																														
	I																														
	I																														
	I																														
	I																														

1-2

IBM

INTERNATIONAL BUSINESS MACHINES CORPORATION

REPORT PROGRAM GENERATOR CALCULATION SPECIFICATIONS

IBM System 360

Date

Program

Programmer

Punching Instruction	Graphic							
	Punch							

Page 03

Program Identification PRO01

Line	Form Type	Control Level (L0-L9, LR)	Indicators: Not	And Not	And Not	Factor 1	Operation	Factor 2	Result Field	Field Length	Decimal Positions	Half Adjust (H)	Resulting Indicators Plus / Compare High 1 > 2	Minus / Low 1 < 2	Equal 1 = 2	Comments
3 4 5	6	7 8	9 10 11	12 13 14	15 16 17	18 19 20 21 22 23 24 25 26 27	28 29 30 31 32	33 34 35 36 37 38 39 40 41 42	43 44 45 46 47 48	49 50 51	52	53	54 55	56 57	58 59	60 61 62 63 64 65 66 67 68 69 70 71 72 73 74
01	C		01			FINTOT	ADD	INVAMT	FINTOT	9	2					
02	C															
03	C															
04	C															
05	C															
06	C															
07	C															
08	C															
09	C															
10	C															
11	C															
12	C															
13	C															
14	C															
15	C															
	C															
	C															
	C															
	C															
	C															

1-3

IBM

INTERNAT NAL BUSINESS MACHINES ORPORATION

REPORT PROGRAM GEI ERATOR OUTPU -FORMAT SPECII ICATIONS

IBM System 360

Date

Program

Programmer

Punching Instruction	Graphic							
	Punch							

Page 04

Program Identification PR061

Line	Form Type	Filename	Type (H/D/T)	Stacker Select	Space Before	Space After	Skip Before	Skip After	Output Indicators	And	And	Field Name	Zero Suppress (Z)	Blank After (B)	End Position in Output Record	Packed Field (P)	Constant or Edit Word	Sterling Sign Position
3 4 5	6	7–14	15	16	17	18	19 20	21 22	Not 23, 24 25	Not 26, 27 28	Not 29, 30 31	32–37	38	39	40–43	44	45–70	71–74
0 1	O	REPORT	H			2	01		1P									
0 2	O		OR						OR									
0 3	O														20		'ACCOUNTS'	
0 4	O														48		'RECEIVABLE REGISTER'	
0 5	O		H			1			1P									
0 6	O		OR						OR									
0 7	O														8		'CUSTOMER'	
0 8	O														54		'LOCATION INVOICE'	
0 9	O														80		'INVOICE INVOICE'	
1 0	O		H			2			1P									
1 1	O		OR						OR									
1 2	O														24		'NUMBER CUSTOMER NAME'	
1 3	O														54		'STATE CITY NUMBER'	
1 4	O														80		'DATE AMOUNT'	
1 5	O																	
	O																	
	O																	
	O																	
	O																	
	O																	

1-4

5-1

INTERNATIONAL BUSINESS MACHINES CORPORATION

Form X24-3352-1

Printed in U.S.A.

IBM REPORT PROGRAM GENERATOR OUTPUT-FORMAT SPECIFICATIONS

IBM System 360

Date ____________

Program ____________

Programmer ____________

Punching Instruction		
Graphic	Z	
Punch	A	

Page 05

Program Identification PR0B1

Line	Form Type	Filename	Type (H/D/T)	Stacker Select	Space Before	Space After	Skip Before	Skip After	Output Indicators (Not)	And (Not)	And (Not)	Field Name	Zero Suppress (Z)	Blank After (B)	End Position in Output Record	Packed Field (P)	Constant or Edit Word	Sterling Sign Position
01	O		D			2			01									
02	O											CUSNO	Z		6			
03	O											CUSNAM			33			
04	O											STATE	Z		39			
05	O											CITY	Z		44			
06	O											INVNO	Z		54			
07	O											INVDAY			66		' 0/ / '	
08	O											INVAMT			80		' , 0. '	
09	O		T		2			01	LR									
10	O											FINTOT			82		'$, , 0. &*'	
11	O														62		'FINAL TOTAL:'	
12	O																	
13	O																	
14	O																	
15	O																	
	O																	
	O																	
	O																	
	O																	
	O																	

LINE DESCRIPTION

FIELD HEADINGS/WORD MARKS

6 Lines Per Inch

IBM 407, 408, 409, 1403, 1404, 1 , and 220

DAILY SALES REGISTER

PRACTICE PROBLEM #2

GLUE

```
*H     6            D A I L Y      S A L E S      R E G I S T E R
       7
 H     8   WEEK OF  XXX XX                               PAGE XXXX
       9           (MONTH)(DAY)
      10
 H    11            ENTRY   CUSTOMER   SALESMAN         SALE
 H    12             DAY     NUMBER     NUMBER         AMOUNT
      13
 D    14              XX      XXXXX       XXX        $XX,XX0.XX
      15
      16
 T(L1)17                           DAY XX SALES     $XXX,XX0.XX *
      18
      19
      20
      21
      22
 T(LR)23   NUMBER OF SALES XXX      TOTAL SALES   $X,XXX,XX0.XX **
```

IBM

[illegible] Business [illegible] Corporation

REPORT PROGRAM GENERATOR FILE DESCRIPTION SPECIFICATIONS

IBM System 360

Date ____________

Program DAILY SALES REGISTER

Programmer JAMES BOND

Punching Instruction	
Graphic	
Punch	

Page 01

Program Identification PROB2

Line (3-5)	Form Type (6)	Filename (7-14)	I/O/U/C (15)	P/S/C/R/T (16)	E (17)	A/D (18)	F/V (19)	Block Length (20-23)	Record Length (24-27)	L/R (28)	Length of Record Address Field (29-30)	K/I (31)	I/D/T (32)	Overflow Indicator (33-34)	Key Field Starting Location (35-38)	Extension Code E/L (39)	Device (40-46)	Symbolic Device (47-52)	Labels S, N, or E (53)	Name of Label Exit (54-59)	Extent Exit for DAM (60-65)	Comments (66-74)
01	F	SALES	I	P			F	80	80								READ40	SYSRDR				
02	F	REGISTER	O				V	132	132					OF			PRINTER	SYSLST				
03	F																					
04	F																					
05	F																					
06	F																					
07	F																					
08	F																					
09	F																					
10	F																					
11	F																					
12	F																					
13	F																					
14	F																					
15	F																					
	F																					
	F																					
	F																					
	F																					
	F																					

2-1

IBM

INTERNATIONAL BUSINESS MACHINES CORPORATION

Printed in U.S.A.

REPORT PROGRAM GENERATOR INPUT SPECIFICATIONS

IBM System 360

Date
Program
Programmer

Punching Instruction	Graphic								
	Punch								

Page 02 (1 2)

Program Identification PROB2 (75 76 77 78 79 80)

Line	Form Type	Filename	Sequence	Number (1-N)	Option (O)	Resulting Indicator	Record Identification Codes												Stacker Select	Packed (P)	Field Location		Decimal Positions	Field Name	Control Level (L1-L9)	Matching Fields or Chaining Fields	Field Record Relation	Field Indicators			Sterling Sign Position
							1				2				3																
							Position	Not (N)	C Z D	Character	Position	Not (N)	C Z D	Character	Position	Not (N)	C Z D	Character			From	To						Plus	Minus	Zero or Blank	
3 4 5	6	7 8 9 10 11 12 13 14	15 16	17	18	19 20	21 22 23 24	25	26	27	28 29 30 31	32	33	34	35 36 37 38	39	40	41	42	43	44 45 46 47	48 49 50 51	52	53 54 55 56 57 58	59 60	61 62	63 64	65 66	67 68	69 70	71 72 73 74
01	I	SALES	AA			01	6	N	Z	-																					
02	I																				6	8	0	SLSMN							
03	I																				9	10	0	ENTRY	L1						
04	I																				11	15	0	CUST							
05	I																				51	57	2	SLSAMT							
06	I		BB			02	6		Z	-																					
07	I																				1	3		MONTH							
08	I																				4	5	0	DAY							
09	I																														
10	I																														
11	I																														
12	I																														
13	I																														
14	I																														
15	I																														
	I																														
	I																														
	I																														
	I																														
	I																														

REPORT PROGRAM GENERATOR FILE DESCRIPTION SPECIFICATIONS

IBM System 360

Date ____________

Program WEEKLY PAYROLL REGISTER

Programmer TOM SWIFT

Punching Instruction	Graphic							
	Punch							

Page 01

Program Identification PR003

Line	Form Type	Filename	File Type I/O/U/C	File Designation P/S/C/R/T	End of File E	Sequence A/D	File Format F/V	Block Length	Record Length	Mode of Processing L/R	Length of Record Address Field	Record Address Type K/I	Type of File Organization I·D/T	Overflow Indicator	Key Field Starting Location	Extension Code E/L	Device	Symbolic Device	Labels S, N, or E	Name of Label Exit	Extent Exit for DAM	Comments
3 4 5	6	7 8 9 10 11 12 13 14	15	16	17	18	19	20 21 22 23	24 25 26 27	28	29 30	31	32	33 34	35 36 37 38	39	40 41 42 43 44 45 46	47 48 49 50 51 52	53	54 55 56 57 58 59	60 61 62 63 64 65	66 67 68 69 70 71 72 73 74
01	F	EARNINGS	I	P		A	F	80	80								READ40	SYSIPT				
02	F	PAYLIST	O				V	132	132					OB			PRINTER	SYSLST				
03	F	SUMMARY	O				F	80	80								READ40	SYSOPT				
04	F																					
05	F																					
06	F																					
07	F																					
08	F																					
09	F																					
10	F																					
11	F																					
12	F																					
13	F																					
14	F																					
15	F																					
	F																					
	F																					
	F																					
	F																					

3-1

IBM

INTERNATIONAL BUSINESS MACHINES CORPORATION

REPORT PROGRAM GENERATOR INPUT SPECIFICATIONS

IBM System 360

Date ____________

Program ____________

Programmer ____________

Punching Instruction	Graphic							
	Punch							

Page (1 2) 02

Program Identification (75 76 77 78 79 80) PROB3

Line	Form Type	Filename	Sequence	Number (1-N)	Option (O)	Resulting Indicator	Record Identification Codes												Stacker Select	Packed (P)	Field Location		Decimal Positions	Field Name	Control Level (L1 L9)	Matching Fields or Chaining Fields	Field-Record Relation	Field Indicators			Sterling Sign Position
							1				2				3																
							Position	Not (N)	C/Z/D	Character	Position	Not (N)	C/Z/D	Character	Position	Not (N)	C/Z/D	Character			From	To						Plus	Minus	Zero or Blank	
3 4 5	6	7 8 9 10 11 12 13 14	15 16	17	18	19 20	21 22 23 24	25	26	27	28 29 30 31	32	33	34	35 36 37 38	39	40	41	42	43	44 45 46 47	48 49 50 51	52	53 54 55 56 57 58	59 60	61 62	63 64	65 66	67 68	69 70	71 72 73 74
0 1	I	EARNINGS	AA			01	80		C	E																					
0 2	I																				14	16	0	DEP	L1	M1					
0 3	I																				17	21	0	SERIAL							
0 4	I																				57	61	2	GROSS							
0 5	I																				62	65	2	INS							
0 6	I																				69	72	2	WTAX							
0 7	I																				73	75	2	UCI							
0 8	I																				76	79	2	MISC							
0 9	I		BB			02	80		C																						
1 0	I																				1	3		MON							
1 1	I																				4	5	0	DATE							
1 2	I																														
1 3	I																														
1 4	I																														
1 5	I																														
	I																														
	I																														
	I																														
	I																														

3-2

IBM

INTERNATIONAL BUSINESS MACHINES CORPORATION

REPORT PROGRAM GENERATOR CALCULATION SPECIFICATIONS

IBM System 360

Date ____________

Program ____________

Programmer ____________

Punching Instruction	Graphic							
	Punch							

Page 03

Program Identification PR003

Line (3-5)	Form Type (6)	Control Level (L0-L9, LR) (7-8)	Indicators: Not (9-11)	Indicators: And Not (12-14)	Indicators: And Not (15-17)	Factor 1 (18-27)	Operation (28-32)	Factor 2 (33-42)	Result Field (43-48)	Field Length (49-51)	Decimal Positions (52)	Half Adjust (H) (53)	Resulting Indicators: Plus / Compare: High 1 > 2 (54-55)	Resulting Indicators: Minus / Compare: Low 1 < 2 (56-57)	Resulting Indicators: Zero / Compare: Equal 1 = 2 (58-59)	Comments (60-74)
01	C		01			GROSS	MULT	.042	FICA	3	2	H				
02	C		01			GROSS	SUB	FICA	NET	5	2					
03	C		01			NET	SUB	INS	NET							
04	C		01			NET	SUB	WTAX	NET							
05	C		01			NET	SUB	UCI	NET							
06	C		01			NET	SUB	MISC	NET					H1	H2	
07	C		01	NH1		DEPNET	ADD	NET	DEPNET	6	2					
08	C	L1				FINNET	ADD	DEPNET	FINNET	8	2					
09	C															
10	C															
11	C															
12	C															
13	C															
14	C															
15	C															
	C															
	C															
	C															
	C															
	C															

B-8

IBM

INTERNATIONAL BUSINESS MACHINES CORPORATION

REPORT PROGRAM GENERATOR OUTPUT-FORMAT SPECIFICATIONS

IBM System 360

[illegible]
Printed in U.S.A.

Date ______

Program ______

Programmer ______

Punching Instruction: Graphic ☐☐☐☐☐☐☐ Punch ☐☐☐☐☐☐☐

Page 04

Program Identification PROB3

Line (3-5)	Form Type (6)	Filename (7-14)	Type (H/D/T) (15)	Stacker Select (16)	Space Before (17)	Space After (18)	Skip Before (19-20)	Skip After (21-22)	Output Indicators (23-25)	And (26-28)	And (29-31)	Field Name (32-37)	Zero Suppress (Z) (38)	Blank After (B) (39)	End Position in Output Record (40-43)	Packed Field (P) (44)	Constant or Edit Word (45-70)	Sterling Sign Position (71-74)
01	O	PAYLIST	H				01	02	02									
02	O	OR							OB									
03	O											MON			4			
04	O											DAY	Z		7			
05	O														95		'P.'	
06	O											PAGE	Z		100			
07	O		D			1			01									
08	O											DEP	Z		5			
09	O											SERIAL	Z		13			
10	O											GROSS			26		' 0. '	
11	O											INS			37		' 0. '	
12	O											FICA			47		'0. '	
13	O											WTAX			58		' 0. '	
14	O											UCI			68		'0. '	
15	O											MISC			80		' 0. '	
16	O											NET			97		' 0. '	
	O																	
	O																	
	O																	
	O																	

Form X20-3336 [illegible]
Printed in U.S.A.

IBM

INTERNATIONAL BUSINESS MACHINES CORPORATION

REPORT PROGRAM GENERATOR OUTPUT-FORMAT SPECIFICATIONS

IBM System 360

Date ____________

Program ____________

Programmer ____________

Punching Instruction	Graphic							
	Punch							

Page 05

Program Identification PRO03

Line	Form Type	Filename	Type (H/D/T)	Stacker Select	Space Before	Space After	Skip Before	Skip After	Output Indicators Not	Output Indicators	And Not	And	And Not	And	Field Name	Zero Suppress (Z)	Blank After (B)	End Position in Output Record	Packed Field (P)	Constant or Edit Word	Sterling Sign Position
3 4 5	6	7–14	15	16	17	18	19 20	21 22	23	24 25	26	27 28	29	30 31	32–37	38	39	40–43	44	45–70	71–74
01	O		T		1	2				L1											
02	O														DEPNET			99		' , 0. &*'	
03	O		T		1					LR											
04	O														FINNET			102		' , 0. &***'	
05	O		T				01	01		LR											
06	O																	60		'END OF JOB'	
07	O	SUMMARY	T	4						L1											
08	O														DEPNET		B	22			
09	O														DEP			16			
10	O														DAY			13			
11	O														MON			11			
12	O																	80		'-'	
13	O		T	4						LR											
14	O																	1		' '	
15	O	* DUMMY CARD PUNCH TO EJECT LAST DATA CARD INTO STACKER.																			
	O																				
	O																				
	O																				
	O																				
	O																				

REPORT PROGRAM GENERATOR FILE DESCRIPTION SPECIFICATIONS

IBM System 360

1 of 4

Date –

Program EMPLOYEE REPORT

Programmer ELBIE JAYE

Punching Instruction: Graphic / Punch

Page 01

PRO85

Line	Form Type	Filename	File Type I/O/U/C	File Designation P/S/C/R/T	End of File E	Sequence A/D	File Format F/V	Block Length	Record Length	Mode of Processing L/R	Length of Record Address Field	Record Address Type K/T	Type of File Organization I D/T	Overflow Indicator	Key Field Starting Location	Extension Code E/L	Device	Symbolic Device	Labels S N or E	Name of Label Exit	Extent Exit for DAM	Comments
3 4 5	6	7 8 9 10 11 12 13 14	15	16	17	18	19	20 21 22 23	24 25 26 27	28	29 30	31	32	33 34	35 36 37 38	39	40 41 42 43 44 45 46	47 48 49 50 51 52				
0 1	F	MASTER	I	P	E	A	F	80	80								READ40	SYSRDR				
0 2	F	DETAIL	I	S	E	A	F	80	80								READ01	SYSIPT				
0 3	F	PRINT	O				V	132	132					OF			PRINTER	SYSLST				
0 4	F																					
0 5	F																					
0 6	F																					
0 7	F																					
0 8	F																					
0 9	F																					
1 0	F																					
1 1	F																					
1 2	F																					
1 3	F																					
1 4	F																					
1 5	F																					
	F																					
	F																					
	F																					
	F																					
	F																					

1-5

IBM

ESS MACHINES CORPORATION

REPORT PROGRAM GENERATOR INPUT SPECIFICATIONS

IBM System 360

2 of 4

Date ______

Program ______

Programmer ______

Punching Instruction

Page 02

Program Identification PROB5

Line	Form Type	Filename	Sequence	Number (1-N)	Option (O)	Resulting Indicator	Record Identification Codes 1 Position	1 Not (N)	1 C/Z/D	1 Character	2 Position	2 Not (N)	2 C/Z/D	2 Character	3 Position	3 Not (N)	3 C/Z/D	3 Character	Stacker Select	Packed (P)	Field Location From	Field Location To	Decimal Positions	Field Name	Control Level (L1-L9)	Matching Fields or Chaining Fields	Field Record Relation	Field Indicators Plus	Field Indicators Minus	Field Indicators Zero or Blank	Sterling Sign Position
3 4 5	6	7-14	15 16	17	18	19 20	21-24	25	26	27	28-31	32	33	34	35-38	39	40	41	42	43	44-47	48-51	52	53-58	59 60	61 62	63 64	65 66	67 68	69 70	71-74
01	I	MASTER	MA			01	80		C	-																					
02	I																				1	5	0	MASNUM		M1					
03	I																				6	6		INIT1							
04	I																				7	7		INIT2							
05	I																				8	30		NAME							
06	I	DETAIL	DA			02	80		C	1																					
07	I																				1	5	0	NUM		M1					
08	I																				6	10	0	AMT1				03			
09	I																				11	15	0	AMT2				04			
10	I																														
11	I																														
12	I																														
13	I																														
14	I																														
15	I																														
	I																														
	I																														
	I																														
	I																														
	I																														

IBM

INTERNA CORPORATION

REPORT PROGRAM GENERATOR CALCULATION SPECIFICATIONS

IBM System 360

Printed in U.S.A.

3 of 4

Date

Program

Programmer

Punching Instruction	Graphic
	Punch

Page 03

Program Identification PROB5

Line (3-5)	Form Type (6)	Control Level (L0-L9, LR) (7-8)	Indicators: Not (9-11)	Indicators: And Not (12-14)	Indicators: And Not (15-17)	Factor 1 (18-27)	Operation (28-32)	Factor 2 (33-42)	Result Field (43-48)	Field Length (49-51)	Decimal Positions (52)	Half Adjust (H) (53)	Resulting Indicators: Plus / Compare: High 1 > 2 (54-55)	Resulting Indicators: Minus / Compare: Low 1 < 2 (56-57)	Resulting Indicators: Compare: Equal 1 = 2 (58-59)	Comments (60-74)
01	C		MR	03	04	AMT1	ADD	AMT2	TOTAL	6	0					
02	C															
03	C															
04	C															
05	C															
06	C															
07	C															
08	C															
09	C															
10	C															
11	C															
12	C															
13	C															
14	C															
15	C															
	C															
	C															
	C															
	C															
	C															

IBM

INTERNATIONAL BUSINESS MACHINES CORPORATION

REPORT PROGRAM GENERATOR OUTPUT-FORMAT SPECIFICATIONS

IBM System 360

4 of 4

Date ____________

Program ____________

Programmer ____________

Punching Instruction: Graphic / Punch

Page 04

Program Identification PROB5

Line	Form Type	Filename	Type (H/D/T)	Stacker Select	Space Before	Space After	Skip Before	Skip After	Output Indicators: Not	Output Indicators	And: Not	And	And: Not	And	Field Name	Zero Suppress	Blank After (B)	End Position in Output Record	Packed Field (P)	Constant or Edit Word	Sterling Sign Position
01	O	PRINT	H			1	01			1P											
02	O		OR							OF											
03	O																	23		'EMPLOYEE EMPLOYEE NAME'	
04	O																	54		'AMOUNT AMOUNT'	
05	O																	64		'TOTAL'	
06	O		H			2				1P											
07	O		OR							OF											
08	O																	7		'NUMBER'	
09	O																	43		'1'	
10	O																	52		'2'	
11	O		D			2				MR		02									
12	O														NUM	Z		7			
13	O														INIT1			11			
14	O														INIT2			13			
15	O														NAME			37			
	O														AMT1			46		' , 0-'	
	O														AMT2			55		' , 0-'	
	O										03		04		TOTAL			64		' , 0'	
	O																				
	O																				

4-9

E026615MSP

Card Electro N

U

Form X24-3347-2 U/M
Printed in U.S.A.

IBM

INTERNATIONAL BUSINESS MACHINES CORPORATION

REPORT PROGRAM GENERATOR FILE DESCRIPTION SPECIFICATIONS

IBM System 360

1 of 4

Date ____________

Program EMPL. EARNINGS MASTER UPDATE

Programmer A. EINSTEIN

Punching Instruction	Graphic							
	Punch							

Page 01

Program Identification PR006

Line (3-5)	Form Type (6)	Filename (7-14)	File Type I/O/U/C (15)	File Designation P/S/C/R/T (16)	End of File E (17)	Sequence A/D (18)	File Format F/V (19)	Block Length (20-23)	Record Length (24-27)	Mode of Processing L/R (28)	Length of Record Address Field (29-30)	Record Address Type K/I (31)	Type of File Organization I/D/T (32)	Overflow Indicator (33-34)	Key Field Starting Location (35-38)	Extension Code E/L (39)	Device (40-46)	Symbolic Device (47-52)	Labels (S, N, or E) (53)	Name of Label Exit (54-59)	Extent Exit for DAM (60-65)	Comments (66-74)
01	F	TAPEIN	I	P		A	F	520	52								TAPE	SYS001				
02	F	CARDS	I	S		A	F	80	80								READ40	SYS002				
03	F	TAPEOUT	O				F	520	52								TAPE	SYS003				
04	F																					
05	F																					
06	F																					
07	F																					
08	F																					
09	F																					
10	F																					
11	F																					
12	F																					
13	F																					
14	F																					
15	F																					
	F																					
	F																					
	F																					
	F																					
	F																					

1-7

IBM

INTERNATIONAL BUSINESS MACHINES CORPORATION

REPORT PROGRAM GENERATOR INPUT SPECIFICATIONS

IBM System 360

2of

Date ____________

Program ____________

Programmer ____________

Punching Instruction	Graphic						
	Punch						

Page Ø2

Program Identification PRØB6

Line	Form Type	Filename	Sequence	Number (1-N)	Option (O)	Resulting Indicator	Record Identification Codes 1: Position	Not (N)	C/Z/D	Character	2: Position	Not (N)	C/Z/D	Character	3: Position	Not (N)	C/Z/D	Character	Stacker Select	Packed (P)	Field Location From	To	Decimal Positions	Field Name	Control Level (L1-L9)	Matching Fields or Chaining Fields	Field-Record Relation	Field Indicators Plus	Minus	Zero or Blank	Sterling Sign Position
01	I	TAPEIN	AA			Ø1																									
02	I																				1	2	Ø	TDEPT		M2					
03	I																				3	5	Ø	TCLOCK		M1					
04	I																				6	25		TNAME							
05	I																				26	34	Ø	TSS							
06	I																				35	41	2	TGROSS							
07	I																				42	47	2	TWTAX							
08	I																				48	52	2	TFICA							
09	I																				1	52		RECORD							
10	I	CARDS	BB			Ø2	8Ø		D	1																					
11	I																				1	2	Ø	CDEPT		M2					
12	I																				3	5	Ø	CCLOCK		M1					
13	I																				62	68	2	CGROSS							
14	I																				69	74	2	CWTAX							
15	I																				75	79	2	CFICA							
	I																														
	I																														
	I																														
	I																														
	I																														
	I																														

REPORT PROGRAM GENERATOR CALCULATION SPECIFICATIONS
IBM System 360

3 of 4

Date

Program

Programmer

Punching Instruction	Graphic							
	Punch							

Page 03

Program Identification PR086

Line	Form Type	Control Level (L0-L9, LR)	Indicators: Not		And: Not		And: Not		Factor 1	Operation	Factor 2	Result Field	Field Length	Decimal Positions	Half Adjust (H)	Resulting Indicators: Plus / High 1 > 2	Minus / Low 1 < 2	Zero / Equal 1 = 2	Comments
01	C			MR		02			TGROSS	ADD	CGROSS	NGROSS	7	2					
02	C			MR		02			TWTAX	ADD	CWTAX	NWTAX	6	2					
03	C			MR		02			TFICA	ADD	CFICA	NFICA	5	2					
04	C																		
05	C																		
06	C																		
07	C																		
08	C																		
09	C																		
10	C																		
11	C																		
12	C																		
13	C																		
14	C																		
15	C																		
	C																		
	C																		
	C																		
	C																		
	C																		

REPORT PROGRAM GENERATOR OUTPUT-FORMAT SPECIFICATIONS

IBM System 360

4 of 4

Date ____________

Program ____________

Programmer ____________

Punching Instruction	
Graphic	
Punch	

Page 04

Program Identification PR066

Line	Form Type	Filename	Type (H/D/T)	Stacker Select	Space Before	Space After	Skip Before	Skip After	Not	Output Indicator	And Not	Output Indicator	And Not	Output Indicator	Field Name	Zero Suppress (Z)	Blank After (B)	End Position in Output Record	Packed Field (P)	Constant or Edit Word	Sterling Sign Position
01	O	TAPEOUT	D							MR		02									
02	O														TDEPT			2			
03	O														TCLOCK			5			
04	O														TNAME			25			
05	O														TSS			34			
06	O														NGROSS		B	41			
07	O														NWTAX		B	47			
08	O														NFICA		B	52			
09	O	TAPEOUT	D							MR		02									
10	O														RECORD			52			
11	O														NGROSS		B	41			
12	O														NWTAX		B	47			
13	O														NFICA		B	52			
14	O		D						N	MR		01									
15	O														RECORD			52			
	O																				
	O																				
	O																				
	O																				
	O																				

ALTERNATE CODING

IBM

INTERNATIONAL BUSINESS MACHINES CORPORATION

REPORT PROGRAM GENERATOR FILE DESCRIPTION SPECIFICATIONS

IBM System 360

Date ____________

Program INVENTORY FILE UPDATE

Programmer N. BONE APARTE

Punching Instruction — Graphic / Punch

Page 01

Program Identification PROG7

Line	Form Type	Filename	I/O/U/C	P/S/C/R/T	E	A/D	F/V	Block Length	Record Length	L/R	Length of Record Address Field	K/I	I D/T	Overflow Indicator	Key Field Starting Location	Extension Code E/L	Device	Symbolic Device	Labels	Name of Label Exit	Extent Exit for DAM	
3 4 5	6	7 8 9 10 11 12 13 14	15	16	17	18	19	20 21 22 23	24 25 26 27	28	29 30	31	32	33 34	35 36 37 38	39	40 41 42 43 44 45 46	47 48 49 50 51 52	53	54		
01	F	* MASTER INVENTORY FILE IS INDEXED-SEQUENTIAL, UPDATED RANDOMLY.																				
02	F	*																				
03	F	INFIL	U	C			F	125	125	R		K	I				DISK11	SYS003	S			CHAINED FILE
04	F	CARDIN	I	P	E		F	80	80							E	READ40	SYSRDR				CHAINING FILE
05	F	CARDOUT	O				F	80	80								READ40	SYSOPT				EXCEPTION FILE
06	F																					
07	F																					
08	F																					
09	F																					
10	F																					
11	F																					
12	F																					
13	F																					
14	F																					
15	F																					
	F																					
	F																					
	F																					
	F																					
	F																					

1-6

IBM Sy. n 360

Date ____________

Program ____________

Programmer ____________

Punching Instruction	Graphic	
	Punch	

Page 02

Program Identification PROB7

Line	Form Type	Record Sequence of the Chaining File	Number of the Chaining Field	From Filename	To Filename	Table Name	Number of Table Entries Per Record	Number of Table Entries Per Table	Length of Table Entry	Packed (P)	Decimal Pos.	Sequence (A/D)	Table Name (Alternating Table)	Length of Table Entry	Packed(P)	Decimal Pos.	Sequence (A/D)	Comments
3 4 5	6	7 8	9 10	11–18	19–26	27–32	33–35	36–39	40–42	43	44	45	46–51	52–54	55	56	57	58–74
0 1	E	CC	C1	CARDIN	INFIL													
0 2	E																	

IBM

INTERNATIONAL BUSINESS MACHINES CORPORATION

REPORT PROGRAM GENERATOR INPUT SPECIFICATIONS 03

IBM System 360

PROB7

Form X24-3350-1 U/M025
Printed in U.S.A.

Line	Form Type	Filename	Sequence	Number (1-N)	Option (O)	Resulting Indicator	Record Identification Codes 1 Position	1 Not (N)	1 C/Z/D	1 Character	2 Position	2 Not (N)	2 C/Z/D	2 Character	3 Position	3 Not (N)	3 C/Z/D	3 Character	Stacker Select	Packed (P)	Field Location From	Field Location To	Decimal Positions	Field Name	Control Level (L1-L9)	Matching Fields or Chaining Fields	Field-Record Relation	Field Indicators Plus	Field Indicators Minus	Field Indicators Zero or Blank	Sterling Sign Position
3 4 5	6	7–14	15 16	17	18	19 20	21–24	25	26	27	28–31	32	33	34	35–38	39	40	41	42	43	44–47	48–51	52	53–58	59 60	61 62	63 64	65 66	67 68	69 70	71–74
0 1	I	INFIL	AA			01	125		D	2																					
0 2	I																				1	8		PART							
0 3	I																				10	36		DESC							
0 4	I																				37	42	0	VENDOR							
0 5	I																				50	53	0	MIN							
0 6	I																				54	58	0	ONHAND							
0 7	I	CARDIN	CC			03	1		D	1																					
0 8	I																				2	9		DTLPAR		C1					
0 9	I																				10	13	0	RECPTS							
1 0	I																				14	17	0	ISSUES							
1 1	I																				18	21	0	RETURN							
1 2	I		BB			02	1		C	*																					

IBM

INTERNATIONAL BUSINESS MACHINES CORPORATION

REPORT PROGRAM GENERATOR CALCULATION SPECIFICATIONS

IBM System 360

Form X24-33[illegible]
Printed in U.S.A.

Date
Program
Programmer

Punching Instruction	
Graphic	
Punch	

Page 04

Program Identification PR087

Line	Form Type	Control Level (L0-L9, LR)	Indicators: Not / And	Indicators: Not / And	Indicators: Not	Factor 1	Operation	Factor 2	Result Field	Field Length	Decimal Positions	Half Adjust (H)	Resulting Indicators: Plus / Compare: High 1 > 2	Resulting Indicators: Minus / Compare: Low 1 < 2	Resulting Indicators: Zero or Blank / Compare: Equal 1 = 2	Comments
3 4 5	6	7 8	9 10 11	12 13 14	15 16 17	18–27	28–32	33–42	43–48	49–51	52	53	54 55	56 57	58 59	60–74
01	C		01	03			Z-ADD	ONHAND	NEWBAL	5	0					
02	C		01	03		NEWBAL	ADD	RECPTS	NEWBAL							
03	C		01	03		NEWBAL	SUB	ISSUES	NEWBAL							
04	C		01	03		NEWBAL	ADD	RETURN	NEWBAL							
05	C		01	03		NEWBAL	COMP	MIN						04	05	
06	C		N01	03			SETON						H1			
07	C															
08	C															
09	C															
10	C															
11	C															
12	C															
13	C															
14	C															
15	C															
	C															
	C															
	C															
	C															
	C															

4-7

Form X24-3352-1
Printed in U.S.A.

IBM

INTERNATIONAL BUSINESS MACHINES CORPORATION

REPORT PROGRAM GENERATOR OUTPUT-FORMAT SPECIFICATIONS

IBM System 360

Date ____________

Program ____________

Programmer ____________

Punching Instruction	Graphic							
	Punch							

Page 05 (1 2)

Program Identification PR087 (75 76 77 78 79 80)

Line	Form Type	Filename	Type (H/D/T)	Stacker Select	Space		Skip		Output Indicators						Field Name	Zero Suppress (Z)	Blank After (B)	End Position in Output Record	Packed Field (P)	Constant or Edit Word	Sterling Sign Position
					Before	After	Before	After	Not		And Not		And Not								
3 4 5	6	7 8 9 10 11 12 13 14	15	16	17	18	19 20	21 22	23	24 25	26	27 28	29	30 31	32 33 34 35 36 37	38	39	40 41 42 43	44	45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70	71 72 73 74
0 1	O	INFIL	D							01		03									
0 2	O														NEWBAL			58			
0 3	O	CARDOUT	D	4						01		03		04							
0 4	O	OR		8						01		03		05							
0 5	O														PART			9			
0 6	O														DESC			36			
0 7	O														VENDOR			42			
0 8	O														DATE			48			
0 9	O									04								1		'B'	
1 0	O									05								1		'E'	
1 1	O																				
1 2	O																				
1 3	O																				
1 4	O																				
1 5	O																				
	O																				
	O																				
	O																				
	O																				
	O																				

5-4

Card Electro Number

IBM

INTERNATIONAL BUSINESS MACHINES CORPORATION

Form X24 347-2 U/M025
Printed U.S.A.

REPORT PROGRAM GENERATOR FILE DESCRIPTION SPECIFICATIONS

IBM System 360

Date ______________

Program EMPLOYEE BENEFITS NOTICE

Programmer G. E. T. SMART

Punching Instruction	
Graphic	
Punch	

Page 01

Program Identification PRO89

Line	Form Type	Filename	File Type I/O/U/C	File Designation P/S/C/R/T	End of File E	Sequence A/D	File Format F/V	Block Length	Record Length	Mode of Processing L/R	Length of Record Address Field	Record Address Type K/I	Type of File Organization I, D/T	Overflow Indicator	Key Field Starting Location	Extension Code E/L	Device	Symbolic Device	Labels (S, N, or E)	Name of Label Exit	Extent Exit for DAM	Comments
3 4 5	6	7–14	15	16	17	18	19	20–23	24–27	28	29–30	31	32	33–34	35–38	39	40–46	47–52	53	54–59	60–65	66–74
0 1	F	BENEFIT	I	T			F	80	80							E	READ40	SYSRDR				
0 2	F	DATECARD	I	P			F	80	80								READ40	SYSRDR				
0 3	F	TAPE	I	S	E		F	460	46								TAPE	SYS002	S			
0 4	F	NOTICE	O				F	132	132								PRINTER	SYSLST				
0 5	F	NOHIT	O				F	80	80								READ40	SYS100				
0 6	F	* BENEFIT TABLE TO BE LOADED IN IBM 2540 AHEAD OF DATE CARD.																				
0 7	F																					
0 8	F																					
0 9	F																					
1 0	F																					
1 1	F																					
1 2	F																					
1 3	F																					
1 4	F																					
1 5	F																					
	F																					
	F																					
	F																					
	F																					
	F																					

I-6

IBM

INTERNATIONAL BUSINESS MACHINES CORPORATION

REPORT PROGRAM GENERATOR FILE EXTENSION SPECIFICATIONS

IBM System 360

Form [illegible]
Printed in U.S.A.

Date

Program

Programmer

Punching Instruction									
Graphic									
Punch									

Page 02

Program Identification PR089

Line 3 4 5	Form Type 6	Record Sequence of the Chaining File 7 8	Number of the Chaining Field 9 10	From Filename 11–18	To Filename 19–26	Table Name 27–32	Number of Table Entries Per Record 33–35	Number of Table Entries Per Table 36–39	Length of Table Entry 40–42	Packed (P) 43	Decimal Pos 44	Sequence (A/D) 45	Table Name (Alternating Table) 46–51	Length of Table Entry 52–54	Packed(P) 55	Decimal Pos 56	Sequence (A/D) 57	Comments 58–74
0 1	E			BENEFIT		TABYRS	10	50	2		0	A	TABINS	5		0		
0 2	E																	
0 3	E																	
0 4	E																	
0 5	E																	
0 6	E																	
0 7	E																	
0 8	E																	
0 9	E																	
1 0	E																	
1 1	E																	
1 2	E																	
1 3	E																	
1 4	E																	
1 5	E																	
	E																	
	E																	
	E																	
	E																	
	E																	

9-2

IBM

INTERNATIONAL BUSINESS MACHINES CORPORATION

REPORT PROGRAM GENERATOR INPUT SPECIFICATIONS

IBM System 360

Date

Program

Programmer

Punching Instruction	
Graphic	
Punch	

Page 03

Program Identification PRO89

							Record Identification Codes														Field Location							Field Indicators			
							1				2				3																
Line	Form Type	Filename	Sequence	Number (1-N)	Option (O)	Resulting Indicator	Position	Not (N)	C/Z/D	Character	Position	Not (N)	C/Z/D	Character	Position	Not (N)	C Z D	Character	Stacker Select	Packed (P)	From	To	Decimal Positions	Field Name	Control Level (L1-L9)	Matching Fields or Chaining Fields	Field Record Relation	Plus	Minus	Zero or Blank	Sterling Sign Position
3 4 5	6	7 8 9 10 11 12 13 14	15 16	17	18	19 20	21 22 23 24	25	26	27	28 29 30 31	32	33	34	35 36 37 38	39	40	41	42	43	44 45 46 47	48 49 50 51	52	53 54 55 56 57 58	59 60	61 62	63 64	65 66	67 68	69 70	71 72 73 74
01	I	TAPE	AA			01																									
02	I																				6	15		FIRST							
03	I																				16	16		INIT							
04	I																				17	31		LAST							
05	I																				36	37	0	YRHIRE							
06	I	DATECARD	BB			02	7		C	D																					
07	I																				1	6	0	DATE							
08	I																				5	6	0	YRNOW							
09	I																														
10	I																														
11	I																														
12	I																														
13	I																														
14	I																														
15	I																														
	I																														
	I																														
	I																														
	I																														
	I																														

9-3

IBM

INTERNATIONAL BUSINESS MACHINES CORPORATION

REPORT PROGRAM GENERATOR CALCULATION SPECIFICATIONS

IBM System 360

Form X24-[illegible]51-1 U/M102
Printed in U.S.A.

Date ________
Program ________
Programmer ________

Punching Instruction: Graphic / Punch

Page 04

Program Identification PROB9

Line	Form Type	Control Level (L0-L9, LR)	Indicators: Not	And: Not	And: Not	Factor 1	Operation	Factor 2	Result Field	Field Length	Decimal Positions	Half Adjust (H)	Resulting Indicators: Plus / Compare: High 1 > 2	Minus / Low 1 < 2	Zero / Equal 1 = 2	Comments
3 4 5	6	7 8	9 10 11	12 13 14	15 16 17	18 19 20 21 22 23 24 25 26 27	28 29 30 31 32	33 34 35 36 37 38 39 40 41 42	43 44 45 46 47 48	49 50 51	52	53	54 55	56 57	58 59	60 61 62 63 64 65 66 67 68 69 70 71 72 73 74
0 1	C		01			YRNOW	SUB	YRHIRE	YEARS	2	0					
0 2	C		01			YEARS	LOKUP	TABYRS	TABINS	5	0				03	
0 3	C		N03				SETON						H1			
0 4	C															
0 5	C															
0 6	C															
0 7	C															
0 8	C															
0 9	C															
1 0	C															
1 1	C															
1 2	C															
1 3	C															
1 4	C															
1 5	C															
	C															
	C															
	C															
	C															
	:															

17-6

IBM

INTERNATIONAL BUSINESS MACHINES CORPORATION

REPORT PROGRAM GENERATOR OUTPUT-FORMAT SPECIFICATIONS

IBM System 360

Form X24-3352-1 U/M 025
Printed in U.S.A.

Date ______

Program ______

Programmer ______

Punching Instruction	Graphic							
	Punch							

Page (1 2) 05

Program Identification (75 76 77 78 79 80) PR089

Line (3 4 5)	Form Type (6)	Filename (7–14)	Type (H/D/T) (15)	Stacker Select (16)	Space Before (17)	Space After (18)	Skip Before (19 20)	Skip After (21 22)	Output Indicators Not (23)	(24 25)	And Not (26)	(27 28)	And Not (29)	(30 31)	Field Name (32–37)	Zero Suppress (Z) (38)	Blank After (B) (39)	End Position in Output Record (40–43)	Packed Field (P) (44)	Constant or Edit Word (45–70)	Sterling Sign Position (71–74)
01	O	NOTICE	H				01	02		03											
02	O														DATE			17		' 0/ / '	
03	O		D			0				03											
04	O														FIRST			10			
05	O														INIT			12			
06	O														LAST			28			
07	O														YEARS	Z		46			
08	O														TABINS			71		'$, 0'	
09	O	NOHIT	D							01	N	03									
10	O														FIRST			15			
11	O														LAST			31			
12	O		T							LR											
13	O																	1		' '	
14	O																				
15	O																				
	O																				
	O																				
	O																				
	O																				
	O																				

5-6

RPG 3 TOTAL LEVELS PROBLEM X10

OBJECTIVES: Three levels of totals
Headings
Group Indication
Detail Printed Report
Editing
Date in heading

INPUT:

Prod. Des.	Invoice No.	 Date	Br.	Sl.	Cty.	St.	Cust. No.	Qty.	Product Grp	 Item	Sls. Amt.	Cost Amt.
1 - 20	21 - 29		30-31	32-33	34-36	37-38	39-43	44-47	48-49	50-52	53-58	59-64

OUTPUT: Detail Printed Report

```
  YOUR COMPANY NAME                          7/16/66

           Prod.        Sales              Cost              Gross
Br.  Slsm  Group        Amount             Amount            Profit

1     1     2         $X,XXX.XX          $X,XXX.XX          $X,XXX.XX
                      $X,XXX.XX          $X,XXX.XX          $X,XXX.XX
                     $XX,XXX.XX*        $XX,XXX.XX*        $XX,XXX.XX*

            3         $X,XXX.XX          $X,XXX.XX          $X,XXX.XX
                      $X,XXX.XX          $X,XXX.XX          $X,XXX.XX
                     $XX,XXX.XX*        $XX,XXX.XX*        $XX,XXX.XX*

                                                          $XXX,XXX.XX**

      2     1         $X,XXX.XX          $X,XXX.XX          $X,XXX.XX
                      $X,XXX.XX          $X,XXX.XX          $X,XXX.XX CR
                      $X,XXX.XX          $X,XXX.XX          $X,XXX.XX
                     $XX,XXX.XX*        $XX,XXX.XX*        $XX,XXX.XX*

            5         $X,XXX.XX          $X,XXX.XX          $X,XXX.XX
                     $XX,XXX.XX*        $XX,XXX.XX*        $XX,XXX.XX*

                                                          $XXX,XXX.XX**

                                                        $X,XXX,XXX.XX***

2     1     3         $X,XXX.XX          $X,XXX.XX          $X,XXX.XX
```

1-3 6-10 13-17 20-35 38-53 56-71 Print positions

Program RPG PROBLEM #X10

Programmer

Punching Instruction	Graphic							
	Punch							

Page 01 Program Identification RPGX10

Line	Form Type	Filename	File Type I/O/U/C	File Designation P/S/C/R/T	End of File E	Sequence A/D	File Format F/V	Block Length	Record Length	Mode of Processing L/R	Length of Record Address Field	Record Address Type K/I	Type of File Organization I D/T	Overflow Indicator	Key Field Starting Location	Extension Code E/L	Device	Symbolic Device	Labels S, N, or E	Name of Label Exit	Extent Exit for DAM	Comments
010	F	CARD	I	P	E	A	F	80	80								READ40	SYSRDR				
020	F	REPORT	O				V	132	132					OF			PRINTER	SYSLST				

REPORT PROGRAM GENERATOR INPUT SPECIFICATIONS

IBM System 360

Date

Program

Programmer

Punching Instruction	Graphic							
	Punch							

Page 02 Program Identification RPGX10

Line	Form Type	Filename	Sequence	Number (1-N)	Option (O)	Resulting Indicator	Record Identification Codes 1 Position	Not (N)	C Z D	Character	2 Position	Not (N)	C Z D	Character	3 Position	Not (N)	C Z D	Character	Stacker Select	Packed P	Field Location From	To	Decimal Positions	Field Name	Control Level (L1-L9)	Matching Fields or Chaining Fields	Field Record Relation	Field Indicators Plus	Minus	Zero or Blank	Sterling Sign Position
010	I	CARD	AA			01	80	N	C	D																					
02	I																				53	58	2	SLSAMT							
03	I																				59	64	2	COST							
04	I																				48	49	0	PROD	L1	M1					
05	I																				30	31	0	BR	L3	M3					
06	I																				32	33	0	SLSM	L2	M2					
07	I		PP			02	80		C	D																					
																								DATE							

X10-I

REPORT PROGRAM GENERATOR CALCULATION SPECIFICATIONS
IBM System 360

2 of 3

Date ____________

Program ____________

Programmer ____________

Punching Instruction	Graphic								
	Punch								

Page Ø3

Program Identification RPGX10

Line	Form Type	Control Level (L0-L9, LR)	Not	Indicators	Not (And)	Indicators	Not (And)	Indicators	Factor 1	Operation	Factor 2	Result Field	Field Length	Decimal Positions	Half Adjust (H)	Resulting Indicators Plus / High 1 > 2	Resulting Indicators Minus / Low 1 < 2	Resulting Indicators Zero / Equal 1 = 2	Comments
01 Ø	C			Ø1					SLSAMT	SUB	COST	GROSS	6	2					
02	C			Ø1					SLSAMT	ADD	MINSL	MINSL	7	2					
03	C			Ø1					COST	ADD	MINCT	MINCT	7	2					
04	C			Ø1					GROSS	ADD	MINGR	MINGR	7	2					
05	C	L1							MINGR	ADD	INTGR	INTGR	8	2					
06	C	L2							INTGR	ADD	MAJGR	MAJGR	9	2					
07	C	L3							MAJGR	ADD	FINGR	FINGR	11	2					
08	C																		
09	C																		
10	C																		
11	C																		
12	C																		
13	C																		
14	C																		
15	C																		
	C																		
	C																		
	C																		
	C																		
	C																		

X10-2

IBM

INTERNATIONAL BUSINESS MACHINES CORPORATION

REPORT PROGRAM GENERATOR CALCULATION SPECIFICATIONS

IBM System/360

Form X24-[illegible]
Printed in U.S.A.

283

Date ____________

Program ____________

Programmer ____________

Punching Instruction								
	Graphic							
	Punch							

Page (1 2): 03

Program Identification (75 76 77 78 79 80): RPGX16

Line (3-5)	Form Type (6)	Control Level (L0-L9, LR) (7-8)	Indicators Not (9-11)	Indicators And Not (12-14)	Indicators And Not (15-17)	Factor 1 (18-27)	Operation (28-32)	Factor 2 (33-42)	Result Field (43-48)	Field Length (49-51)	Decimal Positions (52)	Half Adjust (H) (53)	Resulting Indicators Plus / Compare High 1>2 (54-55)	Resulting Indicators Minus / Compare Low 1<2 (56-57)	Resulting Indicators Zero or Blank / Compare Equal 1=2 (58-59)	Comments (60-74)
01	C		01			SLSAMT	SUB	COST	GROSS	6	2					
02	C		01			SLSAMT	ADD	MINSL	MINSL	7	2					
03	C		01			COST	ADD	MINCT	MINCT	7	2					
04	C		01			GROSS	ADD	MINGR	MINGR	7	2					
05	C	L1				MINGR	ADD	INTGR	INTGR	8	2					
06	C	L2				INTGR	ADD	MAJGR	MAJGR	9	2					
07	C	L3				MAJGR	ADD	FINGR	FINGR	11	2					
08	C															
09	C															
10	C															
11	C															
12	C															
13	C															
14	C															
15	C															
	C															
	C															
	C															
	C															
	C															
	C															

Card Electro Number ____________

IBM

INTERNATIONAL BUSINESS MACHINES CORPORATION

3 of 3

REPORT PROGRAM GENERATOR OUTPUT-FORMAT SPECIFICATIONS

IBM System 360

Date ____________

Program ____________

Programmer ____________

Punching Instruction	Graphic							
	Punch							

Page 04

Program Identification RPGX10

Line	Form Type	Filename	Type (H D T)	Stacker Select	Space Before	Space After	Skip Before	Skip After	Output Indicators Not	Output Indicators	And Not	And	And Not	And	Field Name	Zero Suppress (Z)	Blank After (B)	End Position in Output Record	Packed Field (P)	Constant or Edit Word	Sterling Sign Position
01	O	REPORT	H			2	01			OF											
02	O	OR								02											
03	O																	35		'THRUSH ASSOC'	
04	O																	45		'IATES'	
05	O														DATE			65		' 0/ / '	
06	O		D			1				01											
07	O														SLSAMT			32		'$, 0. '	
08	O														COST			50		'$, 0. '	
09	O														GROSS			71		'$, 0. &CR'	
10	O									L1					PROD	Z		17			
11	O									L2					SLSM	Z		10			
12	O									L3					BR	Z		3			
13	O		T		1	2				L1											
14	O														MINSL		B	33		'$., 0. *'	
15	O														MINCT		B	51		'$, 0. *'	
	O														MINGR		B	69		'$, 0. *'	
	O		T			2				L2											
	O														INTGR		B	70		'$, 0. **'	
	O		T			2				L3											
	O														MAJGR		B	71		'$, , 0. ***'	

X10-3

SYSTEM/360 RPG

APPROACHES TO DISK PROCESSING

There are two classes of disk processing capabilities in RPG:

1. **Single Input Data File.**

   Basically, in this class a report is being drawn from the disk file, either sequentially, or randomly with the help of a Record Address File. In the latter case, the supporting RAF cards supply disk record addresses only and are not data records.

2. **Multiple Input Data Files.**

   This approach may be used for report writing and/or "master" file updating, using either the matching (for sequential processing) or the chaining (for random processing) method. In this class, data from both the disk file and the "transaction" file may be used.

CLASS	TYPE OF ORGANIZATION	TYPE OF PROCESSING	ILLUSTRATION
Single Data File	Sequential	Sequential	
	Indexed Sequential	Sequential	D1
		Sequential between limits (with RAF)	D2
		Random (with RAF)	D3
	Direct	Random (with RAF)	D5
Multiple Data Files	Sequential	Sequential (Matching)	
	Indexed Sequential	Sequential (Matching)	
		Random (Chaining)	D4
	Direct	Random (Chaining)	D6

IBM

INTERNATIONAL BUSINESS MACHINES CORPORATION

REPORT PROGRAM GENERATOR FILE DESCRIPTION SPECIFICATIONS

IBM System 360

1 of 12

Date ILLUS D1

Program INDEXED SEQUENTIAL DISK FILE — SINGLE DATA INPUT

Programmer PROCESS ENTIRE FILE SEQUENTIALLY

Punching Instruction	Graphic							
	Punch							

Page ☐☐ Program Identification ☐☐☐☐☐☐

Line	Form Type	Filename	File Type I/O/U/C	File Designation P/S/C/R/T	End of File E	Sequence A/D	File Format F/V	Block Length	Record Length	L/R	Mode of Processing	Length of Record Address Field	Record Address Type K/I	Type of File Organization I D/T	Overflow Indicator	Key Field Starting Location	Extension Code E/L	Device	Symbolic Device	Labels S, N, or E	Name of Label Exit	Extent Exit for DAM	Comments
0 1	F	DISKIN	I	P	E		F	600	100				K	I		1		DISK11	SYS005	S			
0 2	F	PRINTOUT	O				F	132	132						OF			—	—				
0 3	F																						

Form X24-3350-1
Printed in U.S.A.

IBM

INTERNATIONAL BUSINESS MACHINES CORPORATION

REPORT PROGRAM GENERATOR INPUT SPECIFICATIONS

IBM System 360

Date ILLUS D1

Program

Programmer

Punching Instruction	Graphic							
	Punch							

Page ☐☐ Program Identification ☐☐☐☐☐☐

Line	Form Type	Filename	Sequence	Number (1-N)	Option (O)	Resulting Indicator	Record Identification Codes 1 Position	1 Not (N)	1 C Z D	1 Character	2 Position	2 Not (N)	2 C Z D	2 Character	3 Position	3 Not (N)	3 C Z D	3 Character	Stacker Select	Packed (P)	Field Location From	Field Location To	Decimal Positions	Field Name	Control Level (L1-L9)	Matching Fields or Chaining Fields	Field-Record Relation	Field Indicators Plus	Field Indicators Minus	Field Indicators Zero or Blank	Sterling Sign Position
0 1	I	DISKIN	AA			07	66		C	M																					
0 2	I																				1	5	0	CUSTNO							
0 3	I																			P	16	20	2	BAL							
0 4	I																														

INTERNATIONAL BUSINESS MACHINES CORPORATION

REPORT PROGRAM GENERATOR FILE DESCRIPTION SPECIFICATIONS

IBM System 360

2 of 12

Date ILLUS D2

Program INDEXED SEQUENTIAL - SINGLE DATA INPUT

Programmer PROCESS SEQUENTIALLY - BETWEEN LIMITS

Punching Instruction	Graphic
	Punch

Page ☐☐

Program Identification ☐☐☐☐☐☐☐☐

Line (3-5)	Form Type (6)	Filename (7-14)	I/O/U/C (15)	P/S/C/R/T (16)	E (17)	A/D (18)	F/V (19)	Block Length (20-23)	Record Length (24-27)	L/R (28)	Length of Record Address Field (29-30)	K/I (31)	I/D/T (32)	Overflow Indicator (33-34)	Key Field Starting Location (35-38)	Extension Code E/L (39)	Device (40-46)	Symbolic Device (47-52)	Labels S, N, or E (53)	Name of Label Exit (54-59)	Extent Exit for DAM (60-65)	Comments (66-74)
01	F	DISKIN	I	P			F	600	100	L		K	I		1		~	~	S	NOTE: THE RAF IS NOT		
02	F	RAF	I	R			F	80	80		05					E	~	~		DESCRIBED ON INPUT		
03	F	PRINTOUT	O				F	132	132								~	~		SPECS.		
04	F																					

IBM

INTERNATIONAL BUSINESS MACHINES CORPORATION

REPORT PROGRAM GENERATOR FILE EXTENSION SPECIFICATIONS

IBM System 360

Form X24-3348-1 U M025
Printed in U.S.A.

Date ILLUS D2

Program ____________

Programmer ____________

Punching Instruction	Graphic
	Punch

Page ☐☐

Program Identification ☐☐☐☐☐☐☐☐

Line	Form Type	Record Sequence of the Chaining File	Number of the Chaining Field	From Filename	To Filename	Table Name	Number of Table Entries Per Record	Number of Table Entries Per Table	Length of Table Entry	Packed (P)	Decimal Pos	Sequence (A/D)	Table Name (Alternate Table)	Length of Table Entry	Packed (P)	Decimal Pos	Sequence (A/D)	Comments
01	E			RAF	DISKIN													
02	E																	
03	E																	

IBM

INTER ATIONAL BUSINES HINES CC RPORATION

REPORT PROGRAM GENERATOR FILE DESCRIPTION SPECIFICATIONS

IBM System 360

3 of 12

Date ILLUS D3

Program INDEXED SEQUENTIAL – SINGLE DATA INPUT

Programmer PROCESS RANDOMLY – KEYS SPECIFIED BY R.A.F.

Punching Instruction	Graphic								Page		Program Identification	
	Punch											

Line (3-5)	Form Type (6)	Filename (7-14)	File Type: I/O/U/C (15)	File Designation: P/S/C/R/T (16)	End of File: E (17)	Sequence: A/D (18)	File Format: F/V (19)	Block Length (20-23)	Record Length (24-27)	Mode of Processing: L/R (28)	Length of Record Address Field (29-30)	Record Address Type: K/I (31)	Type of File Organization: I/D/T (32)	Overflow Indicator (33-34)	Key Field Starting Location (35-38)	Extension Code E/L (39)	Device (40-46)	Symbolic Device (47-52)	Labels S, N, or E (53)	Name of Label Exit (54-59)	Extent Exit for DAM (60-65)	Comments (66-74)
01	F	DISKIN	I	P			F	600	100	R		K	I		1		~	~	S			
02	F	RAF	I	R	E		F	80	80		05					E	~	~				
03	F	PRINTOUT	O				F	132	132								~	~				
04	F																					

IBM

INTERNATIONAL BUSINESS MACHINES CORPORATION

REPORT PROGRAM GENERATOR FILE EXTENSION SPECIFICATIONS

IBM System 360

Form X24-3348-1 U 8M025
Printed in U.S.A.

Date ILLUS D3

Program ______

Programmer ______

Punching Instruction	Graphic								Page	1 2	Program Identification	
	Punch											

Line	Form Type	Record Sequence of the Chaining File	Number of the Chaining Field	From Filename	To Filename	Table Name	Number of Table Entries Per Record	Number of Table Entries Per Table	Length of Table Entry	Packed (P)	Decimal Pos	Sequence (A, D)	Table Name (Alternating Table)	Length of Table Entry	Packed (P)	Decimal Pos	Sequence (A, D)	Comments
01	E			RAF	DISKIN													
02	E																	

REPORT PROGRAM GENERATOR FILE DESCRIPTION SPECIFICATIONS

IBM System 360

4 of 12

Date ILLUS D4

Program INDEXED SEQUENTIAL – MULTIPLE DATA INPUTS

Programmer PROCESS RANDOMLY – CHAINING

Punching Instruction | Graphic | Punch

Page ☐ Program Identification ☐

Line	Form Type	Filename	File Type I/O/U/C	File Designation P/S/C/R/T	End of File E	Sequence A/D	File Format F/V	Block Length	Record Length	Mode of Processing L/R	Length of Record Address Field	Record Address Type K/I	Type of File Organization I D T	Overflow Indicator	Key Field Starting Location	Extension Code E/L	Device	Symbolic Device	Labels S, N or E	Name of Label Exit	Extent Exit for DAM	Comments
01	F	DISKIN	U	C			F	600	100	R		K	I		1		~	~	S			CHAINED FILE
02	F	TRANS	I	P	E		F	80	80							E	~	~				CHAINING FILE

FILE EXTENSION SPECIFICATIONS

Line	Form Type	Record Sequence of the Chaining File	Number of the Chaining Field	From Filename	To Filename	Table Name	Number of Table Entries Per Record	Entry	Packed (P)	Decimal Pos	Sequence (A/D)	Table Name (Alternating Table)	Entry	Packed (P)	Decimal Pos	Sequence (A/D)	Comments
01	E	AA	C1	TRANS	DISKIN												

INPUT SPECIFICATIONS

Line	Form Type	Filename	Sequence	Number (1-N)	Option (O)	Resulting Indicator	Record Identification Codes 1 Position	Not (N)	C Z D	Character	2 Position	Not (N)	C Z D	Character	3 Position	Not (N)	C Z D	Character	Stacker Select	Packed (P)	Field Location From	To	Decimal Positions	Name	Control Level (L1-L9)	Matching Fields or Chaining Fields	Field Record Relation	Field Indicators Plus	Minus	Zero or Blank	Sterling Sign Position
01	I	TRANS	AA			06	80		C	T																					
02	I																				1	5		CUSTNU		C1					
03	I																				6	11	2	AMT							
04	I	DISKIN	AB			05	66		C	M																					
05	I																				1	5		CUSTNO							
06	I																				93	100	2	BAL							

774

REPORT PROGRAM GENERATOR CALCULATION SPECIFICATIONS

IBM System 360

5 of 12

Date ILLUS D4

Program (CONT.)

Programmer

Punching Instruction	Graphic							
	Punch							

Page ☐☐

Program Identification ☐☐☐☐☐☐

Line	Form Type	Control Level (L0-L9, LR)	Indicators Not	And Not	And Not	Factor 1	Operation	Factor 2	Result Field	Field Length	Decimal Positions	Half Adjust (H)	Resulting Indicators Plus / Compare High 1 > 2	Minus / Low 1 < 2	Zero / Equal 1 = 2	Comments
01	C		05	06		AMT	ADD	BAL	BAL							
02	C		N05	06			SETON						H1			
03	C															

IBM

INTERNATIONAL BUSINESS MACHINES CORPORATION

REPORT PROGRAM GENERATOR OUTPUT-FORMAT SPECIFICATIONS

IBM System 360

Form X24-3352-1 U/M 025
Printed in U.S.A.

Date

Program

Programmer

Punching Instruction	Graphic							
	Punch							

Page ☐☐

Program Identification ☐☐☐☐☐☐

Line	Form Type	Filename	Type (H/D/T)	Stacker Select	Space Before	Space After	Skip Before	Skip After	Output Indicators Not	And Not	And Not	Field Name	Zero Suppress (Z)	Blank After (B)	End Position in Output Record	Packed Field (P)	Constant or Edit Word	Sterling Sign Position
01	O	DISKIN	D						05	06								
02	O											BAL			100			
03	O																	
04	O																	

D4-9

IBM

INTERNATIONAL BUSINESS MACHINES CORPORATION

REPORT PROGRAM GENERATOR FILE DESCRIPTION SPECIFICATIONS

IBM System 360

6 of 12

Date ILLUS D4.1

Program PROCESSING MULTIPLE

Programmer CHAINED FILES

Punching Instruction | Graphic | Punch

Page

Program Identification

Line (3-5)	Form Type (6)	Filename (7-14)	I/O/U/C (15)	P/S/C/R/T (16)	E (17)	A/D (18)	F/V (19)	Block Length (20-23)	Record Length (24-27)	L/R (28)	Length of Record Address Field (29-30)	K/T (31)	I/D/T (32)	Overflow Indicator (33-34)	Key Field Starting Location (35-38)	Extension Code E/L (39)	Device (40-46)	Symbolic Device (47-52)	Labels S, N, or E (53)	Name of Label Exit (54-59)	Extent Exit for DAM (60-65)	Comments
01	F	TRANS	I	P	E		F	80	80							E	~	~				
02	F	CUSTOMER	U	C			F	200	200	R		K	I		1	E	~	~	S			
03	F	BRANCH	U	C			F	100	100	R		K	I		27		~	~	S			
04	F	INVEN	U	C			F	200	200	R		K	I		1		~	~	S			
05	F	PRINTOUT	O				F	132	132								~	~				
06	F																					

Form X24-3346-1 U/M025
Printed in U.S.A.

IBM

INTERNATIONAL BUSINESS MACHINES CORPORATION

REPORT PROGRAM GENERATOR FILE EXTENSION SPECIFICATIONS

IBM System 360

Date ILLUS D4.1

Program

Programmer

Punching Instruction | Graphic | Punch

Page

Program Identification

Line	Form Type	Record Sequence of the Chaining File	Number of the Chaining Field	From Filename	To Filename	Table Name	Number of Table Entries Per Record	Number of Table Entries Per Table	Length of Table Entry	Packed P	Decimal Pos	Sequence A D	Table Name	Length of Table Entry	Packed P	Decimal Pos	Sequence A D	Comments
01	E	AA	C1	TRANS	INVEN													
02	E	AA	C2	TRANS	CUSTOMER													
03	E	BB	C3	CUSTOMER	BRANCH													

D4.1-1

IBM

INTERNATIONAL BUSINESS MACHINES CORPORATION

REPORT PROGRAM GENERATOR INPUT SPECIFICATIONS

IBM System 360

7 of 12

Date ILLUS D4.1 (CONT.)

Program

Programmer

Punching Instruction	Graphic								
	Punch								

Page ☐☐ Program Identification ☐☐☐☐☐☐

Line	Form Type	Filename	Sequence	Number (1-N)	Option (O)	Resulting Indicator	Record Identification Codes 1 Position	1 Not (N)	1 C/Z/D	1 Character	2 Position	2 Not (N)	2 C/Z/D	2 Character	3 Position	3 Not (N)	3 C/Z/D	3 Character	Stacker Select	Packed (P)	Field Location From	Field Location To	Decimal Positions	Field Name	Control Level (L1-L9)	Matching Fields or Chaining Fields	Field-Record Relation	Field Indicators Plus	Field Indicators Minus	Field Indicators Zero or Blank	Sterling Sign Position
01	I	TRANS	AA			01	80		C	5																					
02	I																				1	6		CUSTNU	L1	C2					
03	I																				7	10	0	QTY							
04	I																				11	16		ITEMT		C1					
05	I	CUSTOMER	BB			02	200		C	C																					
06	I																				1	6		CUSTNM							
07	I																				87	89		BROFFC		C3					
08	I																				41	48	2	BALC							
09	I	BRANCH	CC			03	100		C	B																					
10	I																				27	29		BROFFB							
11	I																				51	58	2	BALB							
12	I	INVEN	DD			04	200		C	I																					
13	I																				1	6		ITEMI							
14	I																				17	20	2	COST							
15	I																				36	40	0	ONHAND							
	I																														
	I																														
	I																														
	I																														
	I																														

D4.1-3

REPORT PROGRAM GENERATOR CALCULATION SPECIFICATIONS

IBM System 360

8 of 12

Date ILLUS D4.1 (CONT.)

Program

Programmer

Punching Instruction: Graphic / Punch

Page

Program Identification

Line	Form Type	Control Level (L0-L9, LR)	Indicators Not	Indicators	And Not	And	And Not	And	Factor 1	Operation	Factor 2	Result Field	Field Length	Decimal Positions	Half Adjust (H)	Resulting Indicators Plus / Compare High 1>2	Minus / Low 1<2	Zero / Equal 1=2	Comments
01	C									SETOF						05			
02	C			01		02		03		SETON						05			
03	C			04		05			QTY	MULT	COST	TOTAL	8	2					
04	C			04		05			ONHAND	SUB	QTY	ONHAND							DECREMNT INV.
05	C			04		05			TOTAL	ADD	BALC	BALC							INCREMNT A/R
06	C			04		05			TOTAL	ADD	BALB	BALB							INC. SALES
07	C			04		05			TOTAL	ADD	TOTALC	TOTALC	9	2					ACCUM SALES BY COST.

OUTPUT-FORMAT SPECIFICATIONS

Line	Form Type	Filename	Type (H/D/T)	Stacker Select	Space Before	Space After	Skip Before	Skip After	Output Indicators Not	Output Indicators	And Not	And	And Not	And	Field Name	Zero Suppress	Blank After (B)	Position in Output Record	Packed Field (P)	Constant or Edit Word	Sterling Sign Position
01	O	CUSTOMER	D							04		05									
02	O														BALC			48			
03	O	BRANCH	D							04		05									
04	O														BALB			58			
05	O	INVEN	D							04		05									
06	O														ONHAND			40			
07	O	PRINTOUT	T							L1											
08	O														CUSTNM			10			
09	O														TOTALC		B	30		' , . 0. '	

IBM

REPORT PROGRAM GENERATOR FILE DESCRIPTION SPECIFICATIONS
IBM System 360

9 of 12

Date ILLUS D5
Program DIRECT ORGANIZATION – SINGLE DATA FILE
Programmer PROCESS RANDOMLY USING RAF.

Punching Instruction: Graphic / Punch
Page
Program Identification

Line	Form Type	Filename	I/O/U/C	P/S/C/R/T	E	A/D	F/V	Block Length	Record Length	L/R	Length of Record Address Field	K/I	I D/T	Overflow Indicator	Key Field Starting Location	Extension Code E/L	Device	Symbolic Device	Labels S, N, or E	Name of Label Exit	Extent Exit for DAM	Comments
0 1	F	DISKIN	I	P			F	100	100	R		K	D						S			
0 2	F	RAF	I	R	E		F	80	80		05					E						
0 3	F																					

INTERNATIONAL BUSINESS MACHINES CORPORATION

Form X24-3348-1 U/M025
Printed in U.S.A.

IBM

REPORT PROGRAM GENERATOR FILE EXTENSION SPECIFICATIONS
IBM System 360

Date ILLUS D5
Program
Programmer

Punching Instruction: Graphic / Punch
Page
Program Identification

Line	Form Type	Record Sequence of the Chaining File	Number of the Chaining Field	From Filename	To Filename	Table Name	Number of Table Entries Per Record	Number of Table Entries Per Table	Length of Table Entry	Packed (P)	Decimal Pos	Sequence (A/D)	Table Name (Alternating Table)	Length of Table Entry	Packed (P)	Decimal Pos	Sequence (A/D)	Comments
0 1	E			RAF	DISKIN	CONVER												
0 2	E																	
0 3	E																	
0 4	E																	

IBM

INTERNATIONAL BUSINESS MACHINES CORPORATION

REPORT PROGRAM GENERATOR CALCULATION SPECIFICATIONS

IBM System 360

Form X24-3351-1 Printed in U.S.A.

10 of 12

Date ILLUS D5

Program (CONT.)

Programmer

Punching Instruction	Graphic							
	Punch							

Page ☐☐ Program Identification ☐☐☐☐☐☐☐

Line 3 4 5	Form Type 6	Control Level (L0-L9, LR) 7 8	Indicators: Not 9 10 11	And: Not 12 13 14	And: Not 15 16 17	Factor 1 18–27	Operation 28–32	Factor 2 33–42	Result Field 43–48	Field Length 49–51	Decimal Positions 52	Half Adjust (H) 53	Resulting Indicators: Plus / Compare High 1 > 2 54 55	Minus / Low 1 < 2 56 57	Zero / Equal 1 = 2 58 59	Comments 60–74
01	C					CONVER	RPGCV		TRKADR	8						CONTD CONTAINS
02	C						KEYCV		KEYFLD	5						INFO FROM RAF.
03	C						MOVE	CONTD	KEYFLD							
04	C															
05	C															
06	C															
07	C															
08	C						ERPGC									
09	C															
10	C															
11	C															
12	C															
13	C															
14	C															
15	C															
	C															
	C															
	C															
	C															
	C															

IB. 1

INTERNATIONAL BUSINESS MACHINES CORPORATION

REPORT PROGRAM GENERATOR FILE DESCRIPTION SPECIFICATIONS

IBM System 360

11 of 12

Date: ILLUS D6

Program: DIRECT ORGANIZATION – MULTIPLE DATA FILES

Programmer: PROCESS RANDOMLY – CHAINING

Punching Instruction: Graphic ☐ Punch ☐

Page ☐☐ Program Identification ☐☐☐☐☐☐

Line	Form Type	Filename	I/O/U/C	P/S/C/R/T	E	A/D	F/V	Block Length	Record Length	L/R	K/I	I/D/T	Key Field Starting Location	Extension Code E/L	Device	Symbolic Device	Labels (S, N, or E)	Name of Label Exit	Extent Exit for DAM	Comments
0 1	F	DISKIN	U	C			F	100	100	R	K	D					S			
0 2	F	TRANS	I	P	E		F	80	80					E						
0 3	F																			

?

IBM

INTERNATIONAL BUSINESS MACHINES CORPORATION

REPORT PROGRAM GENERATOR FILE EXTENSION SPECIFICATIONS

IBM System 360

Form X24-3348-1 U/M 025
Printed in U.S.A.

Date: ILLUS D6

Program ______________________

Programmer ______________________

Punching Instruction: Graphic ☐ Punch ☐

Page ☐☐ Program Identification ☐☐☐☐☐☐

Line	Form Type	Record Sequence of the Chaining File	Number of the Chaining Field	From Filename	To Filename	Table Name	Number of Table Entries Per Record	Number of Table Entries Per Table	Length of Table Entry	Packed (P)	Decimal Pos.	Sequence (A/D)	Table Name (Alternating Table)	Length of Table Entry	Packed (P)	Decimal Pos.	Sequence (A/D)	Comments
0 1	E	AA	C1	TRANS	DISKIN	CONRTN												
0 2	E																	
0 3	E																	

1-74

REPORT PROGRAM GENERATOR CALCULATION SPECIFICATIONS

IBM System 360

120412

Date Illus D6

Program

Programmer

Punching Instruction								
	Graphic							
	Punch							

Page 1 2

Program Identification 75 76 77 78 79 80

Line (3 4 5)	Form Type (6)	Control Level (L0-L9, LR) (7 8)	Indicators: Not (9 10 11)	Indicators: And Not (12 13 14)	Indicators: And Not (15 16 17)	Factor 1 (18–27)	Operation (28–32)	Factor 2 (33–42)	Result Field (43–48)	Field Length (49 50 51)	Decimal Positions (52)	Half Adjust (H) (53)	Resulting Indicators Plus / Compare High 1>2 (54 55)	Resulting Indicators Minus / Compare Low 1<2 (56 57)	Resulting Indicators Zero / Compare Equal 1=2 (58 59)	Comments (60–74)
01	C					CONRTN	RPGCV		TRACK	8						
02	C						KEYCV		CUSTT	5						
03	C															
04	C															
05	C															
06	C															
07	C						ERPGC									
08	C															
09	C															
10	C															
11	C															
12	C															
13	C															
14	C															
15	C															
	C															
	C															
	C															
	C															
	C															

PROGRAMMING EXAMPLES
AND
CLASS ROOM HANDOUTS
FOR
PROGRAMMING LANGUAGE I

SUMMARY OF PL/I STATEMENTS:

1)NON-EXECUTABLE	DECLARE	COMPILE TIME STMTS
	FORMAT	COMMENT
2)PROCESSING	ASSIGNMENT	PROCEDURE
	GO TO	ENTRY
	IF	BEGIN
	DO	END
	CALL	STOP
	RETURN	
3)ASYNCHRONOUS	WAIT	DELAY
	EVENT, PRIORITY	EXIT
	ALLOCATE,FREE	
	ON & REVERT,SIGNAL	
4) I/O	GET & PUT	DISPLAY
	READ & WRITE	LOCATE
	OPEN & CLOSE	DELETE,UNLOCK
5) DECLARE ATTRIBUTES	FIXED,BINARY	(LEVEL NOS.)
	FLOAT,DECIMAL	(DIMENSIONS)
	REAL,COMPLEX	STATIC,AUTOMATIC
	CHARACTER,BIT	CONTROLLED
	LABEL,SYMBOL	ABNORMAL(USES,SETS)
	DEFINED,LIKE	FILE ATTRIBUTES
	POSITION	ENTRY,GENERIC
	ALIGNED,PACKED	SECONDARY
6) PROCEDURE ATTRUBUTES	MAIN	
	RECURSIVE	
	GENERIC	
	TASK,EVENT,PRIORITY	

FILE DESCRIPTION

```
        DECLARF FILE_NAME  FILE ...ATTRIBUTE-1,ATTRIBUTE-2 ... ;
FILE ATTRIBUTES..

  /* FILE USAGE */          *STREAM OR RECORD
  /* FUNCTION */            *INPUT OR OUTPUT OR UPDATE
  /* ACCESS */              *SEQUENTIAL OR DIRECT
  /* BUFFERING */           *BUFFERED OR UNBUFFERED
                            *EXTERNAL OR INTERNAL
  /* PRINT */                PRINT
  /* BACKWARDS */            BACKWARDS
  /* EXCLUSIVE */            EXCLUSIVE
  /* KEYED */                KEYED
  /* ENVIRONMENT */          ENVIRONMENT (..OPTION-LIST..)
       OPTION-LIST..
                             CONSECUTIVE OR INDEXED OR REGIONAL(1)
                                 OR REGIONAL(27 OR REGIONAL(3)
                             F(BLOCKSIZE,RECORDSIZE) OR V(MAXBLKSZ,
                                 MAXRCDSZ) OR U(MAXBLKSIZE)
                             LEAVE
                             BUFFERS (K)
```

OPEN STATEMENT

```
          OPEN FILE (FILE_NAME) ...OPTIONS...

 OPTIONS:
          IDENT (ARGUMENT)
          TITLE (EXPRESSION)
   /* OPTIONAL */
          INPUT OR OUTPUT OR UPDATE
          STREAM OR RECORD
          DIRECT OR SEQUENTIAL
          BUFFERED OR UNBUFFERED
          EXCLUSIVE
          KEYED
          BACKWARDS
          PRINT
          LINESIZE (EXP)
          PAGESIZE (EXP)
```

CLOSE STATEMENT

```
          CLOSE FILE (FILE_NAME) /* OPTIONAL */ IDENT (ARC) ;
```

A AND X USED AS PICTURE/FORMAT CHARACTERS

IN DECLARE PICTURE ITEM-

PIC'AAAAA' OR PIC'(5)A'	ALPHABETIC OR BLANK
PIC'XXXXX' OR PIC'(5)X'	ANY CHARACTERS

IN EDIT-DIRECTED P FORMAT ITEM-

P'AAAAA' OR P'(5)A'	ALPHABETIC OR BLANK
P'XXXXX' OR P'(5)X'	ANY CHARACTERS

AS EDIT-DIRECTED FORMAT ITEM-

A(10)	CHARACTER STRING
X(10)	HORIZONTAL SKIP OVER (INPUT) OR SPACE (OUTPUT)

EDIT DIRECTED I/O

```
1.  PUT EDIT ('PAGE HEADING', 'FIELD HEADING')(PAGE, X(30),A,LINE (4),
    X(10),A ) ;

2.  DCL 1 RECORD, 2 NAME CHAR (20), 2(BALANCE,LIMIT)FIXED (7,2);
    GET EDIT (RECORD) (A(20), 2 P'99999V99');
    PUT EDIT (RECORD)(A(20), 2(X(10), P 'ZZ,ZZZV.99'));

3.  DCL 1 FIXLD (3);
    PUT EDIT(('X('CAT I CAT ')=',X(I)**2)DO I=1 TO 150);
                   ( 6 ( A,F(10,5)),SKIP );

4.  GET EDIT (RECORD) (R (AWAY));            /* REMOTE FORMAT LIST */
    PUT EDIT(RECORD, A, B, C,) (R(AWAY), 3 F(5));
    AWAY: FORMAT  (A, (20),2F(7,2));
```

STREAM INPUT

```
GET  FILE (FILE_NAME) OR STRING(CHAR_STRING_VAR) OR NOTHING FOR SYSIN

     LIST(DATA LIST) OR DATA(DATA_LIST) OR EDIT(DATA_LIST)(FORMAT_LIST)

     COPY   /*OPTIONAL - USUALLY USED AS A DEBUGGING AID */
```

STREAM OUTPUT

```
PUT  FILE (FILE_NAME) OR STRING (CHAR_STRING_VAR) OR NOTHING FOR SYSPRT

     LIST(DATA_LIST) OR DATA(DATA_LIST) OR EDIT(DATA_LIST)(FORMAT_LIST)
```

STREAM OUTPUT PRINT

```
PUT  FILE (FILE_NAME) OR NOTHING FOR SYSPRINT

     LIST(DATA_LIST) OR DATA(DATA LIST) OR EDIT(DATA LIST)(FORMAT_LIST)
```

LIKE ATTRIBUTE

THE NAME DECLARED IS GIVEN THE SAME STRUCTURE AS THE NAME FOLLOWING THE ATTRIBUTE 'LIKE'.

WHEN NECESSARY, LEVEL NUMBERS WILL BE AUTOMATICALLY ADJUSTED.

THE NAME IMMEDIATELY FOLLOWING THE NAME WITH THE LIKE ATTRIBUTE MUST HAVE AN EQUAL OR LOWER LEVEL NUMBER.

```
     DCL 1 A, 2(B, C, D), 1 E LIKE A;

     DCL 1 A, 2(B, C, D), 1 E, 2(B, C, D),
```

```
DCL                                  *            DCL
1 A,                                 *    1 A,
  2 FIELD1,                          *      2 FIELD1,
    3 DTL 1 PIC '$XXV.99',           *        3 DTL1 PIC '$ZZV.99',
    3 DTL2 CHAR (10),                *        3 DTL2 CHAR(10),
  2 FIELD2 BIT (50),                 *      2 FIELD2 BIT(50),
1 X,                                SAME    X,
  2 FIELD1,                          AS     2 FIELD1,
    3 SUBFIELD1 LIKE A.FIELD1,       *        3 SUBFIELD1,
    3 TABLES(3),                     *          4 DTL1 PIC '$ZZV.99',
  2 FIELD2 LIKE A.FIELD1;            *          4 DTL2 CHAR(10),
                                     *        3 TABLES (3),
                                     *      2 FIELD2,
                                     *        3 DTL1 PIC '$ZZV,99',
                                     *        3 DTL2 CHAR (10);
```

DEFINED ATTRIBUTE

FORM: DEFINED BASE-IDENTIFIER POSITION-OPTION

NOTE: THE DEFINED ITEM OCCUPIES SAME STORAGE AS THE BASE IDENTIFIER.

```
    DECLARE LIST CHAR (40),
             ALIST CHAR(10) DEFINED LIST,
             BLIST CHAR(20) DEFINED LIST POSITION(21),
             CLIST CHAR(10) DEFINED LIST POSITION(11);
         /------------------------LIST - 40------------------------/
         /--ALIST - 10-//--CLIST - 10-//--------BLIST - 20---------/

    DECLARE   PSTRING CHAR (110) ,
            1 P DEFINED PSTRING,
              2 Q CHAR(10) ,
              2 R CHAR(100);
    /----------------------------P - 110---------------------------/
    /---Z - 10----//--------------------R - 100--------------------/
    /-------------------------PSTRING - 110------------------------/

    DECLARE A(3,2), X ( 4, 4);
             A - 1   2              X - 11  12  13  14
                 3   4                  15  16  17  18
                 5   6                  21  22  23  24
                                        25  26  27  28
    DECLARE B(2,3) DEFINED A( 2SUB,  (1SUB);
             B - 1  3  5
                 2  4  6

    DECLARE Z(4) DEFINED X ( 1SUB,  1SUB);
```

```
        /*DATA STRUCTURE PICTURE FORMAT */

    DCL
      1 INVENTORY FILE,
        2 LOCATION,
          3 STATE PIC'9',
          3 WAREHOUSE,
            4 FLOOR PIC'XX',
            4 BIN PIC'XXXX',
        2 PART NUMBER PIC '(7)X',
        2 PART NAME PIC '(21)X',
        2 UNIT OF MEASURE PIC'AA',
        2 REORDER LEVEL PIC'99999',
        2 UNIT COST PIC'99V999',
        2 UNIT SALES(3) PIC'(7)9',
        2 BALANCE ON HAND PIC'(6)9',
        2 ASSEMBLY REQS PIC '999',
        2 QTY PER ASSEMBLY PIC'99',
        2 BIN CAPACITY PIC'99999',
        2 CYCLE PIC '99',
        2 QTY SOLD PIC '99999',
        2 TOTAL COST PIC'9999999V99',
```

```
      /* DATA IN INTERNAL FORMAT */

DECLARE
  1 INVENTORY RECORD,
      2 LOCATION,
          3 STATE        FIXED(1),
          3 WAREHOUSE,
              4 FLOOR    FIXED(2),
              4 BIN      FIXED(4),
      2 PART NUMBER      CHAR(7),
      2 PART NAME        CHAR(21),
      2 UNIT OF MEASURE  CHAR(2),
      2 REORDER LEVEL    FIXED(5),
      2 UNIT COST        FIXED(5,3),
      2 UNIT SALES(3)    FIXED(7,2),
      2 BALANCE ON HAND  FIXED(6),
      2 ASSEMBLY REQ     FIXED(2),
      2 QTY PER ASSEM    FIXED(2),
      2 BIN CAPACITY     FIXED(5),
      2 CYCLE            FIXED(2),
      2 QTY SOLD         FIXED(5),
      2 TOTAL COST       FIXED(9,2);
```

```
EXAMPL1:  PROCEDURE;    /* USING RECORD ORIENTED I/O */
   /* SEQUENTIAL BUFFERED INPUT AND SEQUENTIAL BUFFERED OUTPUT */
  DCL PAYROLL FILE INPUT BUFFERED RECORD SEQUENTIAL ENVIRONMENT
    (CONSECUTIVE F(520,52) BUFFERS(2));
  DCL PAYOUT      FILE OUTPUT BUFFERED SEQUENTIAL ENVIRONMENT
    (CONSECUTIVE F(520,52) BUFFERS(2)) RECORD;
  ON ENFILE (PAYROLL) GO TO WRAPUP;
  OPEN FILE (PAYROLL) TITLE ('INDATA'), FILE (PAYOUT)       TITLE
    ('OUTDATA");
  DCL 1 INPUT,
      2 WKNO          PIC  '99',
      2 EMPL_NO       PIC  '(6)X',
      2 TAX_CLASS     PIC  '99',
      2 YTD_GROSS     PIC  '(5)9V99',
      2 YTD_WITH      PIC  '9999V99',
      2 QTR_FICA      PIC  '999V99',
      2 GROSS_PAY     PIC  '(5)9V99',
      2 WITH_TAX      PIC  '999V99',
      2 FICA          PIC  '999V99',
      2 NET           PIC  '(5)9V99',
     1 OUTPUT LIKE INPUT;
  BACK: READ FILE (PAYROLL) INTO (INPUT);
              ..
              .
        OUTPUT = INPUT;
        WRITE FILE (PAYOUT)      FROM (OUTPUT);
        GO TO BACK;
  WRAPUP:           CLOSE FILE (PAYROLL), FILE (PAYOUT);
 END;
```

* * * * * *

```
EXAMP1A:   PROCEDURE;

   /* SEQUENTIAL BUFFERED INPUT AND OUTPUT IN LOCATE MODE */

  DCL PAYROLL FILE INPUT BUFFERED RECORD SEQUENTIAL ENV(CONSECUTIVE
  F(520,52) BUFFERS(2)),  PAYOUT         FILE OUTPUT BUFFERED RECORD
     SEQUENTIAL ENV(CONSECUTIVE F(520,52) BUFFERS(2));
  ON ENDFILE (PAYROLL) GO TO WRAPUP;
  DCL 1 INPUT CTL (PNTIN),
        2 WKNO        PIC '99',
        2 EMPL_NO     PIC '(6)X',
                           .
                           .
                           .
        2 NET         PIC'(5)9V99',
       1 OUTPUT CTL (PNTOUT) LIKE INPUT;
 OPEN FILE (PAYROLL), FILE (PAYOUT);
  LOOP:  READ FILE (PAYROLL) SET (PNTIN);
           .
           .
         LOCATE FILE (PAYOUT) SET (PNTOUT);
         OUTPUT = INPUT,1/*MOVE FROM INPUT TO OUTPUT BUFFER */
         GO TO LOOP;
     WRAPUP:  CLOSE FILE (PAYROLL), FILE (PAYOUT);
 END EXAMP1A ;
```

```
EXAMPL2:  PROCEDURE;
    /* SEQUENTIAL BUFFEREC UPDATE */
  DCL PAYROLL FILE ENVIRONMENT(INDEXED F(520,52) BUFFERS(3));
  DCL 1 INOUT CTL (PTR),          /* DON'T ALLOCATE STORAGE */
        2 WK NO       PIC '99',
        2 EMPL NO     PIC '(6)X',
           .
           .
        2 NET         PIC '(5)9V99';
  ON ENDFILE (PAYROLL) GO TO THROUGH;
  OPEN FILE(PAYROLL)UPDATE SEQUENTIAL BUFFERED RECORD IDENT(USER_HDR);
  LOOP: READ FILE (PAYROLL) SET (PTR);
           .
           .               /* CHANGE INPUT STRUCTURE*/
           .
        REWRITE FILE (PAYROLL);
        GO TO LOOP;
  THROUGH: CLOSE FILE (PAYROLL)IDENT (USER_TRLR);
END EXAMPL2;
```

* * * * * *

```
EXAMPL3: PROCEDURE;

  /* SEQUENTIAL UNBUFFERED INPUT AND OUTPUT*/

 DCL  PAYROLL FILE ENV(F(520,52) CONSECUTIVE),
      PAYOUT      FILE ENV(F(520,52) CONSECUTIVE);
 ON  ENDFILE (PAYROLL) GO TO DONE;
 'C..  1 INPUT,
 DCL     2 WK NO         PIC '99',
         2 EMPL NO       PIC '(6)X',
           .
           .
         2 NET           PIC '(5)9V99',
       1 OUTPUT LIKE INPUT;
 INIT.. OPEN FILE (PAYROLL) INPUT RECORD SEQUENTIAL UNBUFFERED,
        FILE (PAYOUT     )  OUTPUT RECORD SEQUENTIAL UNBUFFERED;
 LOOP:  READ FILE (PAYROLL) INTO (INPUT) EVENT (READIT);
           .
           .
        IF EVENT (READIT) THEN GO TO PROCESS;
           .
           .
        WAIT (READIT);
 PROCESS:
           .
        OUTPUT = INPUT;
        WRITE FILE(PAYOUT) FROM(OUTPUT) EVENT(WRITEIT);
           .
           .
        WAIT (WRITEIT);
        GO TO LOOP;
           .
           .
 DONE:  CLOSE FILE (PAYROLL), FILE (PAYOUT        );
END EXAMPL3;
```

```
EXAMPL4:  PROCEDURE;

    /*  INQUIRY AS TO STATUS OF AIRLINES */

  DCL FLIGHT FILE INPUT DIRECT UNBUFFERED KEYED   ...ENV(INDEXED
                F(400,80)) , FLIGHTNO  CHAR(4) ;
  OPEN FILE (FLIGHT);
  LOOP: GET LIST ( FLIGHTNO, NUMBER_REQ);
        READ FILE (FLIGHT) INTO (MAJ_STRUCT) KEY (FLIGHTNO )
        EVENT (RESERVN);
          .
          .
        WAIT (RESERVN);
        IF SEATSLEFT GE NUMBER_REQ THEN .:
          .
          .
        GO TO LOOP:
END EXAMPL4;
```

* * * * * *

```
EXAMPL5:  PROCEDURE;

    /* DIRECT UPDATE EXCLUSIVE */

  DCL INVENT    FILE ENV(INDEXED F(500,100)) KEYED   ;
  DCL 1 INVEN_DATA,
        2 ITFM NO     PIC '(6)X',
        2 NO ON HAND  PIC '9999',
           .
           .
        2 RECRDER PT PIC '9999';
  OPEN FILE (INVENT    ) DIRECT UPDATE EXCLUSIVE:
  LOOP: GET LIST (ITEM_NUM, ORDER_QTY);
        READ FILE(INVENT)INTO(INVENT_DATA)KEY(ITEM_NUM)EVENT(GETIT);
           .
           .
        WAIT(GETIT);
        IF ORDER_QTY GT NO_ON_HAND THEN DO; UNLOCK FILE (INVENT);
                                               PUT LIST('SORRY ..NO');
                                           GO TO LOOP;
                                           END;
        NO_ON_HAND = NO_ON_HAND - ORDER_QTY;
        PUT LIST ('ORDER CK');
        REWRITE FILE (INVENT    ) FROM (INVEN_DATA) KEY (ITEM_NUM);
        GO TO LOOP;
END EXAMPL5;
```

```
      PROB5: PROC OPTIONS (MAIN); /* SCIENTIFIC PROBLEM */
        X=1:
        TEST:  IF X LT 1.5 THEN Y = X**3 ;
                                ELSE IF X LE 7.5 THEN Y = X**2;
                                                 ELSE IF X LE 10
                                                    THEN Y = X;
                                                    ELSE GO TO A;

               PUT DATA(X,Y);
               PUT SKIP;
               X = X + .5;
               GO TO TEST;
  A: END PROB5;


      PROB11:  PROCEDURE ; /* PAYROLL PROBLEM */
        DCL WK_NO CHAR(2), EMPL_NO CHAR(6), TAX_CLS FIXED(2),
          (YTD_GROSS, GROSS, NET) FIXED(7,2) YTD_WITH FIXED(6,2),
          (FICA,QTR_FICA,WITH_TAX) FIXED(5,2);
        LOOP: GET LIST (WK_NO,EMPL_NO,TAX_CLS,YTD_GROSS,YTD_WITH,
                 QTR_FICA,GROSS);
              IF WK_NO =1 OR WK_NO=14 OR WK_NO=27 OR WK_NO=40
                THEN QTR_FICA = 0;
              IF WK_NO=1 THEN YTD_WITH,YTD_GROSS = 0;
              IF YTD_GROSS GE 6600. THEN FICA = 0;
                ELSE IF YTD_GROSS + GROSS GE 6600.
                       THEN FICA = .042  * (6600 - YTD_GROSS);
                     ELSE FICA = .042  * GROSS;
              IF TAX_CLS * 13 GE GROSS THEN WITH_TAX = 0;
                ELSE WITH_TAX = (GROSS - TAX CLS * 13) * .14;
              NET = GROSS - FICA - WITH_TAX;
              YTD  GROSS = YTD_GROSS + GROSS;
              YTD_WITH = YTD-WITH + WITH-TAX;
              QTR_FICA = QTR FICA + FICA;
              PUT DATA (WITH_TAX,FICA,NET);
              GO TO LOOP;
       END PROB11;
      PROB11:   PROC; /*PAYROLL USING STRUCTURE & RECORD I/O*/
        /* EXAMPLE OR BUFFERED RECORD ORIENTED*/

        DCL  CRDIN FILE RECORD ENV(F(52) BUFFERS(3)),
             CRDPNCH FILE RECORD OUTPUT ENV(F(52) BUFFERS(3));

        DCL 1 PAYROLL,
              2 WK_NO         PIC'99'
              2 EMPLO_NO      PIC' (6)A' ,
              2 TAX_CLASS     PIC'99',
              2 YTD_GROSS     PIC'99999V99',
              2 YTD_WITH      PIC'9999V99',
              2 QTR_FICA      PIC'999V99',
              2 GROSS_PAY     PIC'(5)9V99',
              2 WITH_TAX      PIC'999V99',
              2 FICA          PIC"999V99',
              2 NET           PIC'(5)9V99',
      ON ENFILE(CRDIN) GO TO END:
      BK: READ FILE (CRDIN) INTO (PAYROLL):
          IF WK NO=1 OR WK_NO=14 OR WK_NO=27 OR WK NO=40
               THEN QTR_FICA = 0;
          IF WK_NO=1 THEN YTD_WITH, YTD_GROSS = 0;
          IF YTD GROSS GE 6600 THEN FICA = 0;
           ELSE IF YTD_GROSS + GROSS GE 6600
                    THEN FICA = .042 * (6600 - YTD_GROSS);
                  ELSE FICA = .042 * GROSS;
          IF TAX·CLASS * 13 GE GROSS THEN WITH_TAX = 0;
           ELSE WITH_TAX = (GROSS - TAX_CLASS * 13) * .14;
          NET = GROSS - WITH_TAX - FICA;
          YTD_GROSS = YTD_GROSS + GROSS;
          YTD_WITH = YTD_WITH+WITH_TAX;
          QTR_FICA = QTR_FICA + FICA;
          WRITE FILE (ORDPNCH) FROM (PAYROLL);
          GO TO BK;
END:END PROB11 ;
```

DO VERB

```
1.  LABEL: IF A = B THEN DO; DISPLAY (A);
                             A = 0;         /* SIMPLE DO */
                             END LABEL;

2.  LABEL: DO WHILE(A LT 100);
           A = A+B * DELTA;                 /* DO WHILE */
              .
              .
           END LABEL;

3.  DO INDEX = 1 TO 25;
          .
          .
       END;

4.  DO I = 10 TO 50 BY 10;
          .      /* INDEXED DO */
          .
        END;

5.  DO INDEX = J+3 BY N TO K*M-4;
          .
          .
        END;

6.  DO INDEX = VALUE + 10 TO LIMIT BY -2 WHILE(VALUE LE SUM(ARRAY X));
          ARRAY X(INDEX) =X**2 + Y/B; -;
                .                                /* MULTIPLE RESTRIC*/
                .
          END;

7.  DO X = A BY 3 WHILE(Y LE N);
          Y=X+B(Y-3);
                .
                .
          END;

8.  LOOP: DO INDEX = J TO N/4 WHILE(X LT 0),N TO 100;
               .
               .
          END LOOP:

9.  DO KNT = 1 TO 10, 20 TO 80 BY 5;   /*MULTIPLE SPECIFICATIONS*/
         .
         .
      END:
```

```
10.  /* NESTED IF AND DO STATEMENTS */
     IF YTDFICA LT 277.20
         THEN CALC: DO: TAX = .C42  *GROSS;
                    IF YTDFICA + TAX GT 277.20
                        THEN DO; TAX = 277.20 - YTDFICA;
                                 YTDFICA = 277.20;
                                 END;
                        ELSE YTDFICA = YTDFICA + TAX;
                    END CALC;
         ELSE TAX = C;
```

```
/*  EXAMPLE OF DO WHILE */

SCROOT:  PROC OPTIONS (MAIN); /* SQUARE ROOT  */
DECLARE COUNT FIXED BINARY;
START: GET LIST(VALUE);
     X0 = VALUE ;/*SET FIRST GUESS = VALUE*/
     X1 = X0 * 2;  /* SET SO FOLLOWING DO IS TRUE INITIALLY*/
  LOOP:  DO COUNT=1 TO 50 HWILE (ABS(X1 - X)) GE 1E-6);
        X0 = X1;
        X1 = 0.5 * ( XC + VALUE / X0 );
        END LOOP;
    IF COUNT = 51 THEN DO; PUT LIST('SQUARE ROOT OF ',VALUE,
                           'FAILS TO CONVERGE');
                            GO TO START;  END;
    PUT DATA (VALUE, X1,COUNT);
    GO TO START;
     END SQROOT;
```

ARRAY ARITHMETIC EXAMPLES.:

```
DCL COMSN (6,4) FIXED(2,2), /* 6 SLSMN , 4 ITMS */ TOTAL FIXED,
    PRICE(4) FIXED(3,2), QTY(4) FIXED(3), PAY (6) FIXED (6,2);
EX1: COMSN = COMSN + .01/* UPDATE ENTIRE TABLE*/
EX2: PAY(3) = SUM ( COMSN ( 3,*)*PRICE * QTY ) ;
   /*  COMMISSION  FOR SALESMAN 3 */

EX3:  TOTAL = SUM ( QTY );
EX4:  PUT LIST ( 'THE COMMISSION FOR SLSMN 2 ON ITEM 4 IS'
                COMSN(2,4));
```

ARRAY ARITHMETIC EXAMPLE

```
WAGES:  PROCEDURE;

     /* COMPUTE THE AVERAGE INCOME OF A SAMPLE POPULATION IN 20 AGE
          GROUPS , EACH WITHIN A 5 YEAR RANGE   */

          DECLARE AGE FIXED, (PEOPLE, INCOME) (0:19) FLOAT;
          INCOME, PEOPLE = 0;
          ON ENDFILE(SYSIN) GO TO TOTAL;
PERSON:   GET LIST (AGE, SALARY);
          PEOPLE(AGE/5) = PEOPLE(AGE/5)+1;
          INCOME(AGE/5) = INCOME(AGE/5) + SALARY; GO TO PERSON;
TOTAL:    PUT LIST ('AVERAGE INCOME BY AGE GROUPS') PAGE;
          DO I = 0 TO 19;
                IF PEOPLE(I) NE 0 THEN INCOME(I) = INCOME (I)
                /PEOPLE (I);
                PUT LIST (5*I, INCOME(I), PEOPLE(I)) SKIP;
          END WAGES;
```

SCIENTIFIC EXAMPLE OF SUBPROGRAM FOR MATRIX INVERSION

```
INVERT;   PROCEDURE (A,INV,N) ;
          DECLARE A(N,N), INV(N,N), D (N) FIXED BIN
            DEFINED INV(1SUB,1SUB);
          INV = 0;  D = 1 ;
              DO K = 1 TO N;
              D = A(K,K);
              A(K,*) = A(K,*)/D;
              INV(K,*) = INV(K,*)/D;
                  DO I = 1 TO K-1,K+1 TO N;
                  D = A(I,K);
                  A(I,*)=A(I,*)-D*A(K,*);
                  INV(I,*) = INV(I,*) - D*A(K,*);
              END INVERT;
```

LABEL VARIABLES - PROGRAM SWITCHES

```
1.  DCL SWITH  LABEL INIT (A);
    GO TO SWITCH;          /* BRANCH TO A*/
         .
    A:   .
         .
        SWITCH = B;                              /* SWITCH IS LABEL VARIABLE,
        .                                         A & B ARE LABEL CONSTANTS */
        GO TO SWITCH;
         .
         .
    B:

2.  LABEL ARRAYS.:
    A: DCL PIVOT(4) LABEL INIT (A,B,C,D);
        C: .
           .
        A: .
           .
        B: .
           .
        D: .
           .
           GO TO PIVOT (3); /* GO TO C */
     B: DCL 1(4) LABEL; /* INIT GENERATED BY COMPILER*/
        L(1): .
              .
        L(4): .
              .
        L(2): .
              .
        L(3): .
              .
              I = 4; GO TO L(I);  /* GO TO L(4) */


     DCL SWITCHEROD LABEL  (LABLi,LABL2,LABL3);

        /* TO AID OPTIMIZATION OF THE OBJECT
             PROGRAM  */
```

```
FIRST: PROC OPTIONS(MAIN);            *
  DCL (A(25),BIG) FIXED(5) EXTERNAL;  * /* SUBROUTINES-NO ARGS */
  GET LIST (A);                       * MAXIMUM: PROC;
  CALL MAXIMUM; PUT DATA (BIG);       *   DCL (A(25),BIG) FIXED(5) EXT;
    .                                 *   BIG = A(1);
    .                                 *        DO KNT = 2 TO 25;
  GET EDIT (A) ( 25 F(5));            *        BIG = MAX(A(KNT),BIG);
    .                                 *        END;
    .                                 *   RETURN;
  CALL MAXIMUM; PUT LIST (BIG);       * END MAXIMUM;
    .                                 *
    .                                 *
  STOP; END FIRST;                    *
```

* * * * * * *

```
FIRST: PROC OPTIONS(MAIN);            *
  DCL (X(20),Y(50),BIG) FIXED(5);     *  /* SUBR-ARGS */
  GET LIST (X);                       * MAXIMUM:  PROC(AR,NO,ANS);
  CALL MAXIMUM(X,20,BIG);             *   DCL (AR(*),ANS) FIXED(5);
  PUT LIST (BIG);                     *   ANS = AR(1);
    .                                 *        DO KNT = 2 TO NO;
    .                                 *        ANS = MAX(AR(KNT),ANS);
  GET LIST (Y);                       *        END;
  CALL MAXIMUM(Y,50,BIG);             *   RETURN;
  PUT DATA (BIG);                     * END MAXIMUM;
    .                                 *
    .                                 *
  STOP;  END FIRST;                   *
```

* * * * * * *

```
                                               /* FUNCTION-ARGS */
FIRST: PROC OPTIONS(MAIN);            * MAXIMUM: PROC(AR,NO);
  DCL (GRDS(30),P(40)) FIXED(5),      *  DCL (AR(*),ANS) FIXED(5);
    (RANGE,BIG,SMALL,AVG) FIXED(5);   *  ANS = AR(1);
    .                                 *       DO KNT = 2 TO NO;
  RANGE = MAXIMUM(GRDS,30) -          *       ANS = MAX(AR(KNT),ANS);
            MINIMUM(GRDS,30);         *       END;
    .                                 *  RETURN(ANS);
    .                                 *       /* MULTIPLE ENTRY POINT*/
  BIG = MAXIMUM(P,40);                * MINIMUM: ENTRY(AR,NO);
    .                                 *  ANS = AR(1);
    .                                 *       DO KNT = 2 TO NO;
  SMALL = MINIMUM(P,40);              *       ANS = MIN(AR(KNT),ANS);
    .                                 *       END;
    .                                 *  RETURN(ANS);
  AVG = SUM(P)/40;                    * END MAXIMUM;
    .                                 *
    .                                 *
  PUT DATA (SMALL,AVG,BIG,RANGE);     *
  STOP;  END FIRST;                   *
```

INTERRUPTS

ON INTERRUPTCONDITION SYSTEM OR
SNAP OR
PROGRAMMER DEFINED ACTION

I/O
- ENDFILE(FIELNAME)
- ENDPAGE(FILENAME)
- TRANSMIT(FILENAME)
- UNDEFINEDFILE(FILENAME)
- NAME(FILENAME)
- KEY(FILENAME)
- RECORD(FILENAME)

COMPUTATIONAL
- CONVERSION *
- FIXEDOVERFLOW *
- OVERFLOW *
- UNDERFLOW *
- ZERODIVIDE *
- SIZE **

LIST PROCESSING
- AREA

CHECKOUT
- CHECK(IDENTIFIER LIST) **
- SUBSCRIPTRANGE **

SYSTEM
- FINISH
- ERROR

REFERENCE: APPENDIX 3

* THESE CONDITIONS ARE ALWAYS ENABLED UNLESS DISABLED BY PROGRAM

** THESE CONDITIONS ARE NOT RECOGNIZED UNLESS ENABLED BY PREFIX.
THE REST OF THE CONDITIONS ARE ALWAYS ENABLED

```
1.  INTERRUPTS THAT ALWAYS OCCUR..  UNDERFLOW
                                    OVERFLOW
                                    ZERODIVIDE
                                    FIXEDOVERFLOW
                                    CONVERSION

    EXCEPT WHEN:
          ( NOOVERFLOW, NOCONVERSION): ABC:   PROCEDURE OR BEGIN
                                         OR   A STATEMENT;

2.  INTERRUPTS THAT NEVER OCCUR:    SIZE
                                    SUBSCRIPTRANGE
                                    CHECK(ITEM:: )

    EXCEPT WHEN:
          (SIZE, SUBSCRIPTRANGE): XYZ: PROC OR BEGIN OR A STATEMENT;

3.  INTERRUPT ACTIONS:
          THIS: PROCEDURE;
            ON ENDPAGE(SYSPRINT) BEGIN;  PUT LIST ('FOOTING');
                                 PUT LIST (' ..HEADING.. ') PAGE;
                                 END'
            ON ENDFILE(SYSIN)  BEGIN; PUT LIST('END OF JOB')LINE(LIST)
                                 CLOSE ...FILES... ;
                                 END;
            ON KEY(PAYROLL)      BEGIN;PUT LIST ('NO SUCH EMPLOYEE AS'
                                 CAT EMPLNO); GO TO READ;
                                 END;
            SIGNAL ENDPAGE(SYSPRINT);
                ..
            GET LIST (A,B,C);
                .
                .
            PUT DATA (A,B,C, ANSWER);
            READ: READ FILE (PAYROLL) INTO (STRUCT) KEY (EMPLNO);
                .
          END THIS;
4.  FACILITIES FOR PROGRAM CHECKOUT:
          (CHECK (A,ST1,SUB2)): MAIN: PROC;
               ON CHECK(A) SNAP DISPLAY('A=' CAT A );
               ON CHECK(ST1) SNAP DISPLAY ( 'HIT ST1 AGAIN');
               ON CHECK(SUB2) SNAP DISPLAY ('INVOKED SUB2 AGAIN') ;
                    A = X+Y;
                    ST1.. R = Q**2;
                    CALL SUB2;
          (SUBSCRIPTRANGE): SUB2: PROC;
                                     .
                                     .
                                  END SUB2;
               ON CONDITION (HEY) ... ;
                                     *
                    SIGNAL CONDITION (HEY);

          END MAIN;
```

```
BLOCK: PROCEDURE;  /* EXAMPLE OF PREFIXES */
     .
     .
ON OVERFLOW BEGIN; DCL NUMBOV STATIC INIT (0);
                   NUMBOV = NUMBOV + 1;
                   IF NUMBOV = 100 THEN GO TO OVERR;
                   END;
     .
     .
ON OVERFLOW;
     .
     .
( NOOVERFLOW): BEGIN;
                .
                .
              END;
BLOCK2: PROCEDURE;
     ON OVERFLOW DO; . . .END;
     REVERT OVERFLOW;
END BLOCK2;
ON OVERFLOW SYSTEM;
     .
     .
END BLOCK1;
```

CONDITION BUILT-IN FUNCTIONS

ONFILE.... CHAR. STRING BEING THE NAME OF THE FILE FOR WHICH THE LAST I/O OPERATION WAS PERFORMED.

ONLOC..... CHAR. STRING BEING THE NAME OF PROCEDURE IN WHICH CONDITION AROSE.

ONSOURCE.. CHAR. STRING BEING CONTENTS OF FIELD BEING PROCESSED WHEN THE LAST CONVERSION INTERRUPT OCCURRED.

ONCHAR.... THE ONE CHARACTER THAT CAUSED THE LAST CONVERSION INTERRUPT.

ONKEY .... CHAR. STRING BEING THE VALUE OF THE KEY FOR THE RECORD WHOSE TRANSMISSION CAUSED THE INTERRUPT.

ONCODE.... A BINARY INTEGER INDICATING THE ERROR...
- CONVERSION INTERRUPT
- CONVERSION INTERRUPT WHILE TRANSMIT PENDING
- BUILT-IN FUNCTION ERROR
- CONTROL PROGRAM ERROR
- I/O ERROR

DATAFIELD. CHAR. LENGTH BEING CONTENTS OF DATA FIELD WHICH GAVE RISE TO LAST NAME CONDITION INTERRUPT.

REFERENCE.. APPENDIX 1.

```
MAIN:  PROC  OPTIONS(MAIN);
         ...
  CALL PRINT (A,B) TASK (T2) EVENT (ET2) PRIORITY (1-2);
           .
           .
  IF EVENT (FT2) THEN GO TO DONE;
         ...
  PRIORITY (T2) = 4;
         ...
  WAIT (ET2);
  DONE:  ...
        STOP;
END MAIN;

            PRINT:  PROC (X,Y);
                      ...
              CALL SET_LINE(L1) EVENT(ET3) PRIORITY(1);
                       .
                       .
                       .
              WAIT(ET3);
                      ...
              EXIT;
            END;

                        SET_LINE: PROC(P);
                              .
                              .
                           EXIT;
                        END;
```

* * * * *

```
/*  PROCEDURE ATTRIBUTES */

    RECURSIVE -
 FACTRL:  PROCEDURE (N)FLOAT(10) OPTIONS(RECURSIVE);
      IF N=1  THEN RETURN (1);
         ELSE RETURN (N * FACTRL (N-1) ) ;
END FACTRL;
```

EXAMPLE USING THE BUILTIN LBOUND AND HBOUND FUNCTIONS

```
FIRST: PROC OPTIONS (MAIN);
DCL(P(0:9), Q(20), L(-5:5))
      FIXED(5);

AVC =MAXIMUM(P)/ 10;
A=MAXIMUM(Q)/20-MAXIMUM(L)/11;
STOP;
END FIRST;
```

```
* MAXIMUM: PROC(AR);
* DCL(AR(*),BIG)FIXED(5):
* BIG=AR(LBOUND(AR,1) ) ;
*   DO KNT=LBOUND(AR,1) +1 TO
*          HBOUND(AR,1);
*   BIG =MAX(AR(KNT),BIG); END;
* RETURN (BIG);
* END MAXIMUM;
```

* * * * * *

SUBROUTINE REFERENCE

```
    /* ENTRY ATTRIBUTE AND MULTIPLE RETURNS */

A:  PROCEDURE;
    DCL  (X,Y) FIXED (7,3), R(10), SUB ENTRY (FLOAT,FLOAT,LABEL);
            .
            .
    CALL SUB (X,R,ERROR);
            .
            .
    ERROR:/* SUBROUTINE RETURNS TO THIS STATEMENT ON OCCURRENCE OF
             EXCEPTIONAL CONDITION */
    END A;

SUB:PROC(A,B,C,);
    DCL B(*), C LABEL;
    /* DIMENSIONS OF B MAY BE DEFINED BY CALLING PROCEDURE */
       .
       .
    IF ....  THEN GO TO LAST;
    IF ....  THEN GO TO C;          / * ERROR EXIT */
    LAST: ..... ;     RETURN;         /* NORMAL RETURN */
END SUB;
```

* * * * * *

SUBSTRING OPERATIONS

```
SUBSTR(STRING_NAME, BEGINNING_CHAR_OR_BIT, LENGTH)

        SUBSTR(CHAR_LIST,40,20)
         SUBSTR (CHAR_LIST, I, N+J)
EXAMPLE:
    GIVEN:    DCL NAME CHAR(30); /* WHERE LAST NAME IS 1ST 20 CHAR,
                       FIRST NAME IS NEXT 9, AND INITIAL IS LAST CHAR.
    1. EXTRACT FIRST NAME:
              DCL FIRST CHAR(9):
              FIRST = SUBSTR(NAME, 21, 9);
    2.  INSERT ANOTHER FIRST NAME IN STRING:
              SUBSTR(NAME, 21, 9) = FIRST;
    3.  REARRANGE NAME FOR PRINTING:
              DCL OUTPUT CHAR (34);
              OUTPUT = SUBSTR (NAME,21,9) CAT ' ' CAT SUBSTR(NAME,30)
        /*  (FIRST) (INITIAL) (LAST) */
```

ASYNCHRONOUS OPERATIONS

```
FORM:
    CALL ENTRY_NAME (ARG_1,ARG_2) /*OPTIONAL*/ TASK OR EVENT OR
                                               PRIORITY;
     .
     .
    EXIT ; /* COMPLETE TASK */
```

1. EVENT OPTION -- FORM: EVENT (SCALAR EVENT NAME)
   USE: ABILITY TO TEST COMPLETION OF TEST WITH EVENT PSUEDO VARIABLE
   ABILIT TO WAIT UNTIL EVENT COMPLETES VIA WAIT STATEMENT

2. PRIORITY OPTION -- FORM: PRIORITY (EXPRESSION)
   USE:EXPRESSION DETERMINES PRIORITY OF CALLED TASK IN RELATION TO MOTHER TASK

3. TASK OPTION -- FORM: TASK (SCALAR TASK NAME)
   USE: NAME TASK CREATED FOR PURPOSES OF CHANGING PRIORITY VIA THE PRIORITY PSEUDO VARIABLE

```
A: PROC  OPTIONS (MAIN);

   CALL  B EVENT(ONE);
      .
      .
   WAIT (ONE);

   STOP;
END A;

                    B: PROC;

                        CALL C EVENT (TWO);
                         .
                         .
                        WAIT (TWO);
                         .
                         .
                        EXIT;
                    END B;

                                  C: PROC;

                                      PUT DATA (ANS);
                                        .
                                        .
                                      EXIT;
                                  END C;
```

PROGRAMMING EXAMPLES
AND
CLASS ROOM HANDOUTS
FOR
ASSEMBLY LANGUAGE CODING

ASSEMBLY LANGUAGE CODING

COMMERCIAL INSTRUCTION SET EXAMPLES

```
* EXAMPLE .
* MULTIPLICATION USING DECIMAL ARITHMETIC.
*
* PROBLEM-    XXX.XX * .XXX * XXX
*    WANT ANSWER ROUNDED TO 2 DIGITS TO RIGHT DECIMAL.
*        ANSWER = XXXXXX.XX
*
EXAMPLE  START
         BALR  2,0
         USING BEGIN,2
BEGIN    BC    15,OKAY
A        DS    CL3                 XXXXXS 2 DIGITS TO RIGHT OF dECIMAL
B        DS    CL2                 XXXS   3 DIGITS TO RIGHT OF DECIMAL
C        DS    CL2                 XXXS   0 DIGITS TO RIGHT OF DECIMAL
PROD     DS    CL7
FIVE     DC    P'500'
*                                  PROD AFTER OPERATION PERFORMED
OKAY     ZAP   PROD,A              00000000XXXXXs  2 DIGITS TO RIGHT
         MP    PROD,B              00000XXXXXXXXS  5 DIGITS TO RIGHT
         MP    PROD,C              00XXXXXXXXXXXS  5 DIGITS TO RIGHT
         MVN   FIVE+1(1),PROD+6    GET SIGN FOR ROUNDING
         AP    PROD,FIVE           ROUND
         MVO   PROD,PROD(5)        00000XXXXXXXXS
         END   EXAMPLE
```

```
* EXAMPLE
*
* DIVISION USING DECIMAL ARITHMETIC.
*
* PROBLEM -  DIVIDEND = XXXXXXX, DIVISOR = XX.X
*   WANT QUOTIENT WITH 3 DIGITS TO RIGHT OF DECIMAL.
*
*   RULE   NO.OF DIGITS TO RIGHT OF DECIMAL IN DIVISOR
*        + NO.OF DIGITS TO RIGHT OF DECIMAL IN QUOTIENT
*        = NO.OF DIGITS TO RIGHT OF DECIMAL IN DIVIDEND
*
*   THUS, WANT DIVIDEND = XXXXXXX.0000
*
EXAMPLE  START 0
         BALR  2,0
         USING *,2
         BC    15,SKIP
QREM     DS    OCL8
QUOT     DS    CL6
REM      DS    CL2
DEND     DS    CL4
SOR      DS    CL2
*                                  QREM AFTER OPERATION PERFORMED.
SKIP     ZAP   QREM(6),DEND        0000XXXXXXXS....
         NC    QREM+5(3),=X'F00000'           00000
         MVN   REM+1(1),DEND+3                    S
         DP    QREM,SOR            QQQQQQQQQQQSRRRS
*
         END   EXAMPLE
```

```
* EXAMPLE
*
* DECIMAL ARITHMETIC GROSS-TO-FIRST-NET
*
*
*
* DECIMAL ARITHMETIC GROSS-TO-FIRST-NET
*
LAB2     START X'1B90'
         BALR  9,0
         USING *,9,10
FIRST    L     10,ADCON
         B     BEGIN
ADCON    DC    A(FIRST+4096)       CONSTANTS
REC      DS    OCL80
ENAME    DS    CL15
EMPNO    DS    CL6
TAXCL    DS    CL2
YTDGRS   DS    CL7
YTDWH    DS    CL6
YTDFIC   DS    CL5
GROSS    DS    CL6
         DS    CL30
*
CURWH    DC    PL6'0'
EXAMT    DC    PL5'0'
TXBLGR   DS    PL4
CURFICA  DC    PL7'0'
UNPDFICA DS    CL3
NETPAY   DS    CL4
SEQUER   DC    C'OUT OF SEQUENCE'
SW       DS    CL1
EDIT     DC    X'4020206B2020214B2020'
SAVE     DC    CL6'000000'
HEAD     DC    C'NAME         '
LINE     DS    OCL72
LNO      DS    CL6
         DS    CL2
LNAME    DS    CL15
         DS    CL1
LGROSS   DS    CL10
         DS    CL1
LFEDWH   DS    CL10
         DS    CL1
LFICA    DS    CL7
         DS    CL1
LCURGR   DS    CL10
         DS    CL1
LNPAY    DS    CL7
BEGIN    OI    SW,X'01'
         SK    1
         WLINE HEAD,13
         XC    LINE,LINE
READ     RCARD REC,EOJ
         TM    SW,X'01'
         BO    OFF
         CLC   EMPNO,SAVE
         BNH   SEQ
MOVE     MVC   SAVE,EMPNO
         PACK  TAXCL,TAXCL    PACK DATA
         PACK  YTDGRS,YTDGRS
         PACK  YTDWH,YTDWH
         PACK  YTDFIC,YTDFIC
         PACK  GROSS,GROSS
*
         AP    YTDGRS,GROSS         COMPUTE NEW YTD GROSS
         ZAP   CURWH(5),=P'0'
*
         ZAP   EXAMT,TAXCL          COMPUTE EXEMPT. AMT.
         MP    EXAMT,=P'2800'
*
         ZAP   TXBLGR,GROSS         COMPUTE TXBL. GROSS
         SP    TXBLGR,EXAMT
         BC    12,A30
*
         ZAP   CURWH,TXBLGR         COMPUTE CURWH
         MP    CURWH,=P'14'
         AP    CURWH,=P'50"
         MVN   CURWH+4(1),CURWH+5
*
         AP    YTDWH,CURWH(5)       COMPUTE NEW YTD WHTAX
A30      ZAP   CURFICA,=P'0'        SET CURFICA TO ZERO
         CP    YTDFIC,=P'174.00'    COMPARE YTDFIC TO $174.00
         BNL   B
*
         ZAP   CURFICA,GROSS        COMPUTE CURFICA
         MP    CURFICA,=P'3625'
         AP    CURFICA,=P'50000'
         MVO   CURFICA,CURFICA(4)
*
         ZAP   UNPDFICA,=P'174.00'  COMPUTE UNPAID FICA
         SP    UNPDFICA,YTDFIC
*
         CP    CURFICA,UNPDFICA     COMPUTE CURFICA TO UNPAID FICA
         BNH   A50
         ZAP   CURFICA,UNPDFICA     SET CURFICA TO UNPAID FICA
*
A50      AP    YTDFIC,CURFICA       COMPUTE NEW YTDFIC
*
B        ZAP   NETPAY,GROSS         COMPUTE NEWPAY
         SP    NETPAY,CURFICA
         SP    NEWPAY,CURWH(5)
         MVC   LNO,EMPNO
         MVC   LNAME,ENAME
         MVC   LGROSS,=X'4020206B2020214B2020'
         ED    LGROSS,YTDGRS+3
         MVC   LFEDWH,EDIT
         ED    LFEDWH,YTDWH+2
         MVC   LFICA,=X'402020214B2020'
         ED    LFICA,YTDFIC+2
         MVC   LCURGR,EDIT
         ED    LCURGR,GROSS+2
         MVC   LNPAY,=X'402020214B2020'
         ED    LNPAY,NETPAY+1
         WLINE LINE,72
         XC    LINE,LINE
         B     READ
EOJ      STOP
OFF      NI    SW,X'00'
         B     MOVE
SEQ      XC    LINE,LINE
         MVC   LNO,EMPNO
         MVC   LNAME,SEQUER
         WLINE LINE,72
```

```
* EXAMPLE
*
* LOAN PROBLEM USING DECIMAL ARITHMETIC
*
EXAMPLE  START 0
         BALR  9,0
         USING *,9
         B     BEGIN
INPUT    DS    OCL80
PRIN     DS    CL7
INT      DS    CL4
PAY      DS    CL4
         DS    CL65
OUTPUT   DS    OCL11
ANSWER   DS    CL3
         DC    C' '
         DC    C'MONTHS.'
WORK     DS    CL9
GOTO     DS    CL3
PRINP    DS    CL4
INTP     DS    CL3
PAYP     DS    CL3
CTR      DC    PL3'O'
FIVE     DC    P'5000'
ONE      DC    P'1'
TWELVE   DC    P'12'
HEAD     DC    C'DECIMAL - NAME'
MESS     DC    C'DONE'
BEGIN    SK    1
         WLINE HEAD,14
READ     RCARD INPUT,EOJ
         PACK  PRINP,PRIN
         PACK  INTP,INT
         PACK  PAYP,PAY
         SP    CTR,CTR
AGAIN    ZAP   WORK,PRINP
         MP    WORK,INTP
         DP    WORK,TWELVE
         MVN   FIVE+2(1),WORK+6
         AP    WORK(7),FIVE
         MVO   WORK(7),WORK(5)
         MVO   WORK(7),WORK(6)
         ZAP   GOTO,PAYP
         SP    GOTO,WORK+4(3)
         SP    PRINP,GOTO
         AP    CTR,ONE
         CP    GOTO,WORK+4(3)
         BH    END
         B     AGAIN
END      UNPK  ANSWER,CTR
         OI    ANSWER+2,X'FO'
         WLINE OUTPUT,13
         B     READ
EOJ      MVI   OUTPUT,C' '
         MVC   OUTPUT+1(12),OUTPUT
         MVC   OUTPUT(4),MESS
         WLINE OUTPUT,13
         STOP
         END   EXAMPLE
```

```
* EXAMPLE
*
* EDIT AND MARK.
*
EXAMPLE  START 0
         BALR  9,0
         USING *,9
         BC    15,BEGIN
*
PAT      DC    X'4020206B2020214B2020'
TOTAL    DS    CL4
WOUT     DS    CL10

BEGIN     .
          .    ACCUMULATE TOTAL
          .
         MVC   WOUT,PAT
         LA    1,WOUT+7
         EDMK  WOUT,TOTAL
         SH    1,=H'1'
         MVI   0(1),C'$'
         END   EXAMPLE
```

```
* EXAMPLE
*
* TRANSLATE AND TEST
*   REPLACE ALL QUESTION MARKS WITH BLANKS.
*   AFTER GET HIT, REINITIALIZE THE TRT.
*
*
EXAMPLE  START 0
         BALR  9,0
         USING *,9
         BC    15,BEGIN
STMT     DS    CL50
TABLE    DC    XL111'0'
         DC    X'40'
         DC    XL144'0'
*
BEGIN    MVI   SCAN+1,49           INITIALIZE TRANSLATE AND TEST
         LA    1,STMT-1
SCAN     TRT   1(,1),TABLE
         BC    6,HIT
CONT     .
         .
         .
HIT      STC   2,0(0,1)            REPLACE QUESTION MARK WITH BLANK
         BC    2,CONT
*
         LA    3,STMT+48           COMPUTE NEW LENGTH
         SR    3,1
         STC   3,SCAN+1
         BC    15,SCAN
*
         END   EXAMPLE
```

```
* EXAMPLE
*
* EXECUTE
*
* USE EX AND TRT TO REPLACE ALL QUESTION MARKS IN STMT WITH BLANKS.
* AFTER GET HIT REINITIALIZE THE NEW LENGTH & STARTING POINT.
*
EXAMPLE  START 0
         BALR  9,0
         USING *,9
         B     BEGIN
TABLE    DC    XL111'0'            CONSTANTS
         DC    X'40'
         DC    XL144'0'
STMT     DS    CL50
SCAN     TRT   1(0,1),TABLE
*
BEGIN    LA    1,STMT-1            INITIALIZE REGISTER 1
AGAIN    LA    3,STMT+48           COMPUTE LENGTH
         SR    3,1
*
         EX    3,SCAN
*
         BC    8,DONE
         STC   2,0(0,1)            REPLACE QUESTION MARK WITH BLANK.
         BC    2,DONE              HIT ON LAST BYTE IN STMT.
         BC    15,AGAIN
DONE     .
         .
         .
         END   EXAMPLE
```

```
* EXAMPLE
*
* TRANSLATE AND REPLACE
*
* WANT TABLE TO TRANSLATE NUMERIC DATA  IN ZONED FORMAT TO BCD.
*
EXAMPLE  START 0
         BALR  9,0
         USING *,9
         BC    15,BEGIN
*
MESSAGE  DS    CL15                CONTAINS DATA IN ZONED NUMERIC.
TABLE    DC    X'0A010203040506070809'
*                                  CONTAINS NUMERIC DATA IN BCD.
*
BEGIN     .
          .
          .
         TR    MESSAGE,TABLE-240   CHARACTER ADJUST BY DECIMAL VALUE
          .                          WHICH EQUALS THE LOWEST POSSIBLE
          .                          BINARY VALUE EXPECTED IN ARGUMENT.
          .                          IE. ZONED ZERO = 11110000 = 240
         END   EXAMPLE
```

ASSEMBLY LANGUAGE CODING

GENERAL PURPOSE REGISTER EXAMPLES

```
* EXAMPLE
*
* DOUBLE INDEXING
*
* WANT TO ADD FULL WORD TO 2ND, 4TH, AND 6TH COLUMNS IN TABLE.
*   TABLE - 10 COL - 15 ROWS - 150 FULL WORDS
*
EXAMPLE  START 0
         BALR  12,0
         USING *,12
         B     SETUP
*
ROWREG   DC    A(TABLE)                INDEX - REG 5
         DC    F'40'                   INCREMENT - REG 6
         DC    A(TABLW+14*40)          LIMIT - REG 7
*
COLREG   DC    F'4"                    INDEX - REG 9
         DC    F'8'                    INCREMENT - REG 10
         DC    F'20                    LIMIT - REG 11
*
CON     DC     F'39515'
*
SETUP    LM    5,7,ROWREG
RLOOP    LM    9,11,COLREG
CLOOP    L     3,CON                   ADD FULL WORD NN TABLE
         A     3,0(9,5)
         ST    3,0(9,5)
         BXLE  9,10,CLOOP
         BXLE  5,6,RLOOP
         .
         .
         .
         END   EXAMPLE
```

```
* EXAMPLE
*
* ADDRESS MODIFICATION USING BASE REGISTER IN SS OR RS FORMAT.
*
* WANT SUM OF 100 VALUES WHICH ARE IN THE PACKED FORMAT AND LOCATED IN
*   CONSECUTIVE LOCATIONS.
*
EXAMPLE  START 0
VALUES   DS    100CL5
         BALR  9,0
         USING *,9
         B     SETUP
*
TOTAL    DS    PL10                 CONSTANTS
OUTPUT   DS    CL10
REGS     DC    F'5'                 INCREMENT
         DC    A(VALUES+99*5)       LIMIT
         DC    A(VALUES)            INDEX
*
SETUP    LM    12,14,REGS
         ZAP   TOTAL,=P'0'
*
PROC     AP    TOTAL,0(5,14)
         BXLE  14,12,PROC
         MVC   OUTPUT,TOTAL
*
         END   EXAMPLE
```

```
* EXAMPLE
*
* ADDRESS MODIFICATION USING BASE REGISTER IN SS OR RS FORMAT.
*
* WANT TO ADD CORRESPONDING ELEMENTS FROM 5 DIFFERENT TABLES AND
*   PLACE RESULT IN 6TH TABLE.  DATA IS IN PACKED FORMAT.
*
EXAMPLE  START 0
T1       DS    100CL5
T2       DS    100CL5
T3       DS    100CL5
T4       DS    100CL5
T5       DS    100CL5
TSUMS    DS    100CL10
*
         BALR  2,0
         USING *,2
         B     SETUP
*
REGS     DC    A(TSUMS)              INSEC - REG 7
         DC    F'10'                 INCREMENT - REG 8
*
         DC    A(T1)                 INDEX - REG 9
         DC    F'5'                  INCREMENT - REG 10
         DC    A(T1+99*5)            LIMIT - REG 11
*
SETUP    LM    7,11,REGS
*
LOOP     ZAP   0(10,7),=P'0'         CLEAR TSUMS
         AP    0(10,7),0(5,9)        ADD CORRESPONDING ELEMENTS FROM
         AP    0(10,7),500(5,9)        T1 TO T5 INTO TSUMS.
         AP    0(10,7),1000(5,9)
         AP    0(10,7),1500(5,9)
         AP    09L0,7),2000(5,9)
*
         BXLE  7,8,*+4               INCREMENT REG 7 BY 10
         BXLE  9,10,LOOP
DONE     .
         .
         .
         END   EXAMPLE
```

```
* EXAMPLE
*
* LOGICAL SHIFTS
*
* HAVE WORD WHICH CONTAINS THE FOLLOWING INFORMATION IN BINARY FORM -
*    YEAR (3 BITS), STUDENT ID (6 BITS), CLASS CODE (8 BITS),
*    GRADE (7 BITS), RANK (8 BITS).  INFORMATION AS DEPICTED
*    IS STORED LEFT TO RIGHT IN THE WORD.
* WANT STUDENT ID, GRADE, AND RANK IN PACKED FORMAT.
*
EXAMPLE  START 0
         BALR  9,0
         USING *,9
         BC    15,BEGIN
WORD     DS    F
WORK     DS    3D
*
BEGIN    L     3,WORD                CONVERT STUDENT ID TO PACKED FORMAT.
         SR    2,2
         SLL   3,3
         SLDL  2,6
         CVD   2,WORK
*
         SR    2,2                   CONVERT GRADE TO PACKED FORMAT.
         SLL   3,8
         SLDL  2,7
         CVD   2,WORK+8
*
         SR    2,2                   CONVERT RANK TO PACKED FORMAT.
         SLDL  2,8
         CVD   2,WORK+16
*
         END   EXAMPLE
```

```
* EXAMPLE
*
* BRANCH ON COUNT
*
* WANT ADDRESS OF RIGHT MOST PERIOD IN STATEMENT.
*
EXAMPLE  START 0
         BALR  9,0
         USING *,9
         B     BEGIN
STMT     DS    CL30
ADDR     DS    F
ADSTMT   DC    A(STMT+29)
*
BEGIN    L     6,ADSTMT                 STARTING POINT
         LA    7,30                     COUNT
*
TEST     CLI   0(6),C'.'
         BE    HIT
         BCTR  6,0                      DECREASE ADDRESS
         BCT   7,TEST                   DECREASE COUNT AND BRANCH NOT ZERO.
NOHITS   .
         .     ROUTINE FOR NO HITS
         .
HIT      ST    6,ADDR
         .
         .
         .
         END   EXAMPLE
```

```
* EXAMPLE
*
* BRANCH INDEX LOW OR EQUAL
*
* WANT AVERAGE OF 365 FULL WORDS WHICH ARE IN CONSECUTIVE LOCATIONS.
*
EXAMPLE  START 0
         BALR  9,0
         USING *,9
         B     SETUP
*
DATA     DS    365F                      CONSTANTS
AVG      DS    F
C365     DC    F'365'
*
SETUP    LA    2,4                       INCREMENT
         LA    3,364*4                   LIMITER
         SR    4,4                       INDEX
         SR    0,0                       CLEAR ACCUMULATOR
*
SUM      A     0,DATA(4)                EXECUTE ADD 365 TIMES.
         BXLE  4,2,SUM
*
         SRDA  0,32                     DIVIDE SUM BY 365.
         D     0,C365
         ST    1,AVG
*
         END   EXAMPLE
```

```
* EXAMPLE
*
* BRANCH INDEX HIGH
*
* WANT AVERAGE OF 365 FULL WORDS WHICH ARE IN CONSECUTIVE LOCATIONS.
*
EXAMPLE  START 0
         BALR  9,0
         USING *,9
         B     SETUP
*
DATA     DS    365F                      CONSTANTS
AVG      DS    F
C365     DC    F'365'
*
SETUP    LA    2,4                       INCREMENT
         LA    3,365*4                   LIMITER
         SR    4,4                       INDEX
         SR    0,0                       CLEAR ACCUMULATOR
*
PROC     BXH   4,2,DONE
         A     0,DATA-4(4)
         B     PROC
*
DONE     SRDA  0,32
         D     0,C365
         ST    1,AVG
*
         END   EXAMPLE
```

```
* EXAMPLE
*
* PROBLEM 42
*
EXAMPLE  START 0
         BALR  2,0
         USING *,2
         BC    15,SKIP
*
RECORD   DS    0CL80
RECORDID DS    CL40                 RESERVE CORE FOR RECORD 1
NEWBAL   DS    CL8
OLDBAL   DS    CL8
ONORD    DS    CL8
REC      DS    CL8
RISSUES  DS    CL8
         DS    9CL80                RESERVE CORE FOR LAST 9 RECORDS
*
SKIP     LA    3,RECORD             INITIALIZE
         LA    4,80
         LA    5,RECORD+720
*
* SOLUTION 1
*
*
AGAIN    ZAP   40(8,3),48(8,3)      COMPUTE NEWBAL
         AP    40(8,3),64(8,3)
         SP    40(8,3),72(8,3)
*
         ZAP   56(8,3),56(8,3)      TEST ON ORDER
*
         BZ    ORDER
         BXLE  3,4,AGAIN
         .
         .
         .
*
* SOLUTION 2
*
*
         USING RECORD,3
AGAIN    ZAP   NEWBAL,OLDBAL
         AP    NEWBAL,REC
         SP    NEWBAL,RISSUES
*
         ZAP   ONORD,ONORD
         DROP  3
*
         BZ    ORDER
         BXLE  3,4,AGAIN
         .
         .
         .
         END   EXAMPLE
```

ASSEMBLY LANGUAGE CODING
FLOATING POINT EXAMPLES

```
* EXAMPLE
*
* FLOATING POINT - FIXED POINT INTEGER TO FLOATING POINT INTEGER.
*
* HAVE FIXED POINT NO. IN LOCATION NUMBER.
* WANT NO. IN FLOATING POINT FORM.IN FPR 0.
*
EXAMPLE  START 0
         BALR  10,0
         USING *,10
         B     START
*                                 CONSTANTS
NUMBER   DC    F'-2'
TEMP     DS    E
SMASK    DC    X'80000000'
X6       DC    X'46000000'
*
START    L     5,NUMBER
         LR    4,5
         N     4,SMASK            SEPARATE SIGN BIT & LEAVE IN GPR 4.
         LPR   3,5                LOAD ABSOLUTE VALE INTO GPR 3.
         OR    3,4                PUT BACK SIGN BIT.
         O     3,X6               TAG ON CHARACTERISTIC OF 6
         ST    3,TEMP
         LE    0,TEMP
         AE    0,X6               NORMALIZE THE NUMBER.
         .
         .
         .
         END   EXAMPLE
```

```
* EXAMPLE
*
* FLOATING POINT - FLOATING POINT INTEGER TO FIXED POINT INTEGER.
*
*
* HAVE FLOATING POINT NUMBER (19.8) IN LOCATION NUMBER.  (CHAR + OR = 6
* WANT INTEGER (19) IN FIXED POINT INTEGER FORM IN GPR 1
*
EXAMPLE  START 0
         BALR  9,0
         USING *,9
         B     BEGIN
*                                  CONSTANTS
NUMBER   DC    E'.255E+2'
TEMP     DS    E
XZERO    DC    X'80FFFFFF'         MASK TO ELIMINATE CHARACTERISTIC.
X6       DC    X'46000000'         FLOATING POINT ZERO WITH CHAR. OF 6
NSIGN    DC    X'7FFFFFFF'         MASK TO ELIMINATE SIGN.
*
*                                  S-CHAR-FRACTION  REGISTER
BEGIN    LE    4,NUMBER            .  42  198000      4 - FPR
         AU    4,X6                .  46  000019      4 - FPR
*                                     INTEGER
         STE   4,TEMP
         L     1,TEMP              .  46  000019      1 - GPR
         N     1,XZERO             .  00  000019
                                      BLANK CHAR
                                      LEAVE SIGN
         LTR   1,1                 TEST SIGN
         BC    10,NOTNEG
*
         N     1,NSIGN             REMOVE SIGN.
         LNR   1,1                 TAKE 2'S COMPLEMENT
NOTNEG   .
         .
         .
         END   EXAMPLE
```

```
* EXAMPLE
*
* FLOATING POINT - SEPARATE INTEGER & FRACTION OF FLOATING POINT NO.
*
* HAVE FLOATING POINT NUMBER (19.8) IN LOCATION NUMBER
*    25.5 BASE 10 = 19.8 TO BASE 16
*
* WANT TO CREATE 2 FLOATING POINT NUMBERS.
     INTEGER (19) IN LOCATION XINTEGER
     FRACTION (8) IN LOCATION FRACTION
*
* NOTE - EXPONENT OF NUMBER IS LESS THAN OR EQUAL TO SIX
         SIGN OF FRACTION = SIGN OF NUMBER
         NUMBER = FRACTION + INTEGER
*
EXAMPLE  START O
         BALR  3,0
         USING *,3
         B     SEPARATE
*
NUMBER   DC    E'25.5'             CONSTANTS
XINTEGER DS    E
FRACTION DS    E
         DC    OD
X6       DC    X'4600000000000000' FLOATING POINT ZERO WITH CHAR. OF 6
*                                    (BIN)(HEX)
*                                  S-CHAR-FRACTION  IN   REGISTER
SEPARATE SDR   6,6                 0  00  00000000000000  6
         LDR   2,6                 0  00  00000000000000  2
         LE    6,NUMBER            0  42  19800000000000  6
*
         AW    6,X6                0  46  00001980000000  6
                                     INTEGER IN 1ST AND
                                     FRACTION IN 2ND HALF
                                     OF FPR 6
*
         LER   2,6                 0  46  00001900000000  2
                                     LOAD INTEGER
*
         SDR   6,2                 0  40  80000000000000  6
                                      FRACTION
*
         STE   2,XINTEGER
         STE   6,FRACTION
         .
         .
         .
         END   EXAMPLE
```

```
* EXAMPLE
*
* FIXED POINT ARITHMETIC - GROSS TO FIRST NET
*
EXAMPLE   START 0
          BALR  9,0
          USING *,9
          B     BEGIN
NWH       DS    D                         DEFINE CONSTANTS
NFICA     DS    D
NET       DS    D
AVG       DS    D
NGROSS    DS    D
K27720    DC    H'27720'
ENAME     DS    CL15
EMPNO     DS    CL6
TAXCL     DS    CL2
YTDGRS    DS    CL7
YTDWH     DS    CL6
YTDFIC    DS    CL5
GROSS     DS    CL6
BEGIN     .
          .     INPUT ROUTINES
          .
          PACK  NGROSS,YTDGRS             COMPUTE NEW GROSS
          CVB   0,NGROSS
          PACK  NGROSS,GROSS
          CVB   2,NGROSS
          AR    0,2
          CVD   0,NGROSS
*
          SR    1,1                       SET CURWH TO ZERO
*
          PACK  NWH,TAXCL                 COMPUTE EXEMPT. AMT.
          CVB   0,NWH
          MH    0,=H'2800'
*
          LR    3,2                       COMPUTE TXBL GROSS
          SR    3,0
          BC    12,A30
*
          LR    1,3                       COMPUTE CURWH
          MH    1,=H'14'
          AH    1,=H'50'
          SR    0,0
          D     0,=F'100'
*
          PACK  NWH,YTDWH                 COMPUTE NEW YTD WHTAX
          CVB   0,NWH
          AR    0,1
          CVD   0,NWH
*
A30       SR    5,5                       SET CURFICA TO ZERO
*
          PACK  NFICA,YTDFIC              COMPARE YTDFIC TO $277.20
          CVB   7,NFICA
          CH    7,K27720
          BC    10,B
*
          LR    5,2                       COMPUTE CURFICA
          MH    5,=H'42'
          A     5,=F'500'
```

```
         SR     4,4
         D      4,=F'1000'
*
         LH     4,K27720
         SR     4,7
*
         CR     5,4                   COMPARE CURFICA TO UNPAID FICA
         BC     12,A50
*
         LR     5,4                   SET CURFICA TO UNPAID FICA
*
A50      AR     7,5                   COMPUTE NEW YTDFIC
         CVD    7,NFICA
*
B        LR     7,2                   COMPUTE NETPAY
         SR     7,5
         SR     7,1
         CVD    7,NET
*
         LR     1,2                   COMPUTE AVG HRLY WAGE
         SR     0,0
         D      0,=F'88'
         CH     0,=H'44'
         BC     4,CON
         AH     1,H'1'
CON      CVD    1,AVG
         .
         .      EDIT AND OUTPUT ROUTINES
         .
         END    EXAMPLE
```

```
* EXAMPLE
LAB1     START X'1B90'
         BALR  9,0
         USING *,9
         B     BEGIN
ZONE     DC    X'F0'
INPUT    DS    0CL80
PRIN     DS    CL7
INT      DS    CL4
PAY      DS    CL4
         DS    CL65
OUTPUT   DS    0CL11
ANSWER   DS    CL3
         DC    C' '
         DC    C'MONTHS.'
WORK     DS    D
ONE      DC    F'1'
HALFAD   DC    F'5000'
TENTH    DC    F'10000'
TWELVE   DC    F'12'
HEAD     DC    C'FIXED POINT - NAME'
MESS     DC    C'DONE'
BEGIN    SK    1
         WLINE HEAD,
READ     RCARD INPUT,EOJ
         PACK  WORK,PRIN            CONVERT CONSTANTS TO BINARY
         CVB   2,WORK
         PACK  WORK,INT
         CVB   3,WORK
         PACK  WORK,PAY
         CVB   4,WORK
         SR    5,5                  CLEAR COUNTER
         LM    6,8 ONE
AGAIN    LR    11,2                 INTEREST TIMES PRIN.
         MR    10,3
         D     10,TWELVE            MONTHLY INTEREST
         AR    11,7                 ADD 5000
         SR    10,10
         DR    10,8                 DROP 4 DIGITS
         LR    10,4                 PAYMENT TO REG. 10
         SR    10,11                SUB. INT. FROM PAY.
         SR    2,10                 SUB. AMT. FROM PRIN.
         AR    5,6                  ADD ONE TO COUNTER
         CR    10,11                COMPARE AMT. APPLIED TO PRIN. TO INT.
         BH    DONE
         B     AGAIN
DONE     CVD   5,WORK               PRINT COUNTER
         UNPK  ANSWER,WORK
         MVZ   ANSWER+2(1),ZONE
         WLINE OUTPUT,11
         B     READ
EOJ      MVI   OUTPUT,C' '
         MVC   OUTPUT+1(10),OUTPUT
         MVC   OUTPUT(4),MESS
         WLINE OUTPUT,11
         STOP
         END   LAB1
```

ASSEMBLY LANGUAGE CODING
INPUT/OUTPUT CONTROL SYSTEMS
FOR
CARD & PRINTER FILES

```
* EXAMPLE
*
* CARD TO PRINT (WITHOUT CARRIAGE CONTROL OR POCKET SELECTION).
*
*
EXAMPLE  START 0
CARD     DTFCD BLKSIZE=40,                                             -
               DEVADDR=SYSIPT,                                         -
               DEVICE=2540,                                            -
               EOFADDR=LSTCD,                                          -
               IOAREA1=CARDIN,                                         -
               MODNAME=CDLOGIC1,                                       -
               RECFORM=FIXUNB,                                         -
               TYPEFLE=INPUT
PRINT    DTFPR BLKSIZE=40,                                             -
               DEVADDR=SYSLST,                                         -
               DEVICE=1403,                                            -
               IOAREA1=CARDIN,                                         -
               MODNAME=PRLOGIC1,                                       -
               RECFORM=FIXUNB
CDLOGIC1 CDMOD RECFORM=FIXUNB,TYPEFLE=INPUT,DEVICE=2540
PRLOGIC1 PRMOD RECFORM=FIXUNB
HERE     CSECT
         BALR  11,0
         USING *,11
         OPEN  CARD,PRINT
NEXT     GET   CARD
         PUT   PRINT
         B     NEXT
LSTCD    CLOSE CARD,PRINT
         EOJ
CARDIN   DS    CL40
         LTORG
         END   HERE
```

```
* EXAMPLE
*
* CARD TO PRINT (USING CNTRL AND PRTOV MACROS WITH A HEADING ROUTINE).
*
*
EXAMPLE  START 0
CARD     DTFCD BLKSIZE=80,DEVADDR=SYSIPT,DEVICE=2540,EOFADDR=LSTCD,
               IOAREA1=N1,MODNAME=CDLOGIC1,RECFORM=FIXUNB,
               TYPEFLE=INPUT
PRINT    DTFPR BLKSIZE=80,CONTROL=YES,DEVADDR=SYSLST,DEVICE=1403,
               IOAREA1=OUT1,MODNAME=PRLOGIC3,PRINTOV=YES,
               RECFORM=FIXUNB
CDLOGIC1 CDMOD RECFORM=FIXUNB,TYPEFLE=INPUT,DEVICE=2540
PRLOGIC3 PRMOD RECFORM=FIXUNB,CONTROL=YES,PRINTOV=YES
HERE     CSECT
         BALR  11,0
         USING *,11
         OPEN  CARD,PRINT
         BAL   14,HDGRT
NEXT     GET   CARD
         MVC   OUT1,N1
         PUT   PRINT
         PRTOV PRINT,12,HDGRT
         B     NEXT
HDGRT    ST    14,HOLD             STORE IOCS REGS.
         CNTRL PRINT,SK,1
            .
            .
         PUT   PRINT               PRINT FIRST HEADING LINE.
         CNTRL PRINT,SP,1
            .
            .
         PUT   PRINT               PRINT SECOND HEADING LINE.
            .
            .
         CNTRL PRINT,SK,2          SKIP TO FIRST BODY LINE.
         L     14,HOLD             RESTORE IOCS REGS.
         BR    14
LSTCD    CLOSE CARD,PRINT
         EOJ
HOLD     DS    F
N1       DS    CL80
OUT1     DS    CL80
*
         LTORG
         END   HERE
```

```
* EXAMPLE
*
* CARD TO PRINT (STACKER SELECTION - CTLCHR - PAGE OVERFLOW).
*
*
EXAMPLE  START 0
CARD     DTFCD BLKSIZE=80,                                             -
               DEVADDR=SYSIPT,                                         -
               DEVICE= 2540.                                           -
               EOFADDR=LSTCD,                                          -
               IOAREA1=N1,                                             -
               MODNAME=CDLOGIC1,                                       -
               RECFORM=F1XUNB,                                         -
               SSELECT=3,                                              -
               TYPEFLE=INPUT
PRINT    DTFPR BLKSIZE=81,                                             -
               CTLCHR=YES,                                             -
               DEVADDR=SYSLST,                                         -
               DEVICE=1403,                                            -
               IOAREA1=OUT1,                                           -
               MODNAME=PRLOGIC2,                                       -
               PRINTOV=YES,                                            -
               RECFORM=FIXUNB
CDLOGIC1 CDMOD RECFORM=FIXUNB,TYPEFLE=INPUT,DEVICE=2540
PRLOGIC2 PRMOD RECFORM=FIXUNB,CTLCHR=YES,PRINTOV=YES
HERE     CSECT
         BALR  11,0
         USING *,11
         B     NEXT1
N1       DS    CL80
OUT1     DS    0CL81
PRINTAR  DC    X'11'
         DS    CL80
NEXT1    OPEN  CARD,PRINT
NEXT     GET   CARD
         MVC   OUT1+1(80),N1
         PUT   PRINT
         PRTOV PRINT,12
         B     NEXT
LSTCD    CLOSE CARD,PRINT
         EOJ
         LTORG
         END   HERE
```

* EXAMPLE .

PROBLEM 01. CARD TO PUNCH.

REQUIREMENTS-

1. READ CARDS FROM SYSIPT (2540).
2. READ ONLY 30 COLUMNS.
3. CARDS WITH CODE '2' IN COL.1 TYPE ON 1052.
4. SELECT CODE '2' CARDS INTO POCKET 2.
5. SELECT ALL OTHER CARDS INTO POCKET 3.
6. PUNCH CARDS ON SYS002.
7. ALL CARDS READ ARE PUNCHED.
8. 1052 IS SYSLOG.

```
*   EXAMPLE    `.
*
*   SOLUTION  PROB. 1.
*
*
EXAMPLE   START 0
CARD      DTFCD BLKSIZE=30,CONTROL=YES,DEVADDR=SYSIPT,DEVICE=2540,        -
                EOFADDR=LSTCD,IOAREA1=N1,RECFORM=FIXUNB,TYPEFLE=INPUT,  -
                MODNAME=CDLOGIC2
PNCH      DTFCD BLKSIZE=30,CRDERR=RETRY,DEVADDR=SYS002,DEVICE=2540,       -
                IOAREA1=N1,MODNAME=PLOGIC1,RECFORM=FIXUNB,               -
                TYPEFLE=OUTPUT
LOG       DTFCN BLKSIZE=30,DEVADDR=SYSLOG,IOAREA1=N1,RECFORM=FIXUNB,      -
                TYPEFLE=OUTPUT
CDLOGIC2  CDMOD RECFORM=FIXUNB,CONTROL=YES,TYPEFLE=INPUT,DEVICE=2540
PLOGIC1   CDMOD RECFORM=FIXUNB,TYPEFLE=OUTPUT,DEVICE=2540,CRDERR=RETRY
HERE      CSECT
          BALR  11,0
          USING *,11
          OPEN  CARD,PNCH
NEXT      GET   CARD
          CLI   N1,C'2'
          BE    LIST
          CNTRL CARD,PS,3
THERE     PUT   PNCH
          B     NEXT
LIST      CNTRL CARD,PS,2
          PUT   LOG
          B     THERE
LSTCD     CLOSE CARD,PNCH
          EOJ
N1        DS    CL30
          LTORG
          END   HERE
```

```
* EXAMPLE
*
* USING TWO I/O AREAS WITH WORK AREA APPROACH.
*
*
         REPRO
         PHASE EXAMPLE6,S
EXAMPLE  START 0
CARD     DTFCD BLKSIZE=80,DEVADDR=SYSIPT,DEVICE=2540,EOFADDR=LSTCD,
               IOAREA1=N1,IOAREA2=N2,RECFORM=FIXUNB,TYPEFLE=INPUT,
               WORKA=YES,MODNAME=CDLOGIC3
PRINT    DTFPR BLKSIZE=80,DEVADDR=SYSLST,DEVICE=1403,IOAREA1=OUT1,
               IOAREA2=OUT2,RECFORM=FIXUNB,WORKA=YES,MODNAME=PRLOGIC4
CDLOGIC3 CDMOD RECFORM=FIXUNB,TYPEFLE=INPUT,WORKA=YES,DEVICE=2540,
               IOAREA2=YES
PRLOGIC4 PRMOD RECFORM=FIXUNB,IOAREA2=YES,WORKA=YES
N1       DS    CL80
N2       DS    CL80
OUT1     DS    CL80
OUT2     DS    CL80
HERE     CSECT
         BALR  11,0
         USING *,11
         OPEN  CARD,PRINT
GET1     GET   CARD,WORK1
         PUT   PRINT,WORK1
         B     GET1
LSTCD    CLOSE CARD,PRINT
         EOJ
WORK1    DS    CL80
         LTORG
         END   HERE
```

```
* EXAMPLE    .
*
* TWO I/O AREAS WITH IOREG APPROACH.
*
*
         REPRO
         PHASE EXAMPLE8,S
EXAMPLE  START 0
CARD     DTFCD BLKSIZE-30,DEVADDR=SYSIPT,DEVICE=2540,EOFADDR=LSTCD,
               IOAREA1=N1,IOAREA2=N2,IOREG=(3),RECFORM=FIXUNB,
               TYPEFLE=INPUT,MODNAME=CDLOGIC4
PRINT    DTFPR BLKSIZE=40,DEVADDR=SYSLST,DEVICE=1403,IOAREA1=OUT1,
               IOAREA2=OUT2,IOREG=(4),RECFORM=FIXUNB,MODNAME=PRLOGIC5
CDLOGIC4 CDMOD RECFORM=FIXUNB,TYPEFLE=INPUT,IOAREA2=YES,DEVICE=2540
PRLOGIC5 PRMOD RECFORM=FIXUNB,IOAREA2=YES
START    CSECT
         BALR  11,0
         USING *,11
         USING DUM1,3
         USING DUM2,4
         OPEN  CARD,PRINT
GET1     GET   CARD
         MVC   SERNO,EMPNO
         MVC   NAMEOUT,NAMEIN
         MVC   AMTOUT,AMTIN
         PUT   PRINT
         B     GET1
LSTCD    CLOSE CARD,PRINT
         EOJ
N1       DS    CL30
N2       DS    CL30
OUT1     DC    4CL10' '
OUT2     DC    4CL10' '
DUM1     DSECT
NAMEIN   DS    CL20
AMTIN    DS    CL5
EMPNO    DS    CL5
DUM2     DSECT
SERNO    DS    CL5
         DS    CL5
NAMEOUT  DS    CL20
         DS    CL5
AMTOUT   DS    CL5
START    CSECT
         LTORG
         END   START
```

```
* EXAMPLE
*
* ILLUSTRATION OF HOW DUMMY SECTION LABELS ASSEMBLE (NO CORE ASSIGNED).


     SOURCE STATEMENTS                  ASSEMBLER ASSIGNED VALUES
               .
               .                  DISPLACEMENT  BASE REG.  LENGTH ATTRIBUTE
               .
CARDIN    DSECT
CODE      DS     CL1                    0           2           1
          DS     CL9
EMPNO     DS     CL5                   10           2           5
          DS     CL5
NAME      DS     CL20                  20           2          20
          DS     CL40
PROG1     CSECT                   NOTE - THE STATEMENTS OF A DUMMY SECTION
          BALR   11,0                    ASSEMBLE AS IF THE DSECT BEGAN IN
          USING  *,11                    STORAGE POSITION ZERO (RELATIVE
          USING  CARDIN,2                TO ZERO).
               .
               .
               .
```

ASSEMBLY LANGUAGE CODING
INPUT/OUTPUT CONTROL SYSTEMS
FOR
MAGNETIC TAPE FILES

```
* EXAMPLE
*
*  DTFMT FOR FIXUNB OUTPUT TAPE WITHOUT LABLES.
*
*
NEWMSTR  DTFMT BLKSIZE=186,
               DEVADDR=SYS005,
               FILABL=NO,
               IOAREA1=OUT1,
               IOAREA2=OUT2,
               MODNAME=TPLOGIC2,
               RECFORM=FIXUNB,
               TYPEFLE=OUTPUT,
               IOREG=(7),          ONE OR OTHER OF THESE
               WORKA=YES           ENTRIES - NOT BOTH.
*
TPLOGIC2 MTMOD RECFORM=FIXUNB,TYPEFLE=OUTPUT,WORKA=YES
```

```
* EXAMPLE
*
*  DTFMT FOR FIXUNB INPUT TAPE WITHOUT LABELS.
*
*
OLDMSTR  DTFMT BLKSIZE=205,                                            -
               DEVADDR=SYS005,                                         -
               EOFADDR=NOTAPE,                                         -
               FILABL=NO,                                              -
               IOAREA1=TPIN1,                                          -
               IOAREA2+TPIN2,                                          -
               MODNAME=TPLOGIC1,                                       -
               RECFORM=FIXUNB,                                         -
               TYPEFLE=INPUT,                                          -
               IOREG=(7),          ONE OR OTHER OF THESE               -
               WORKA=YES           ENTRIES - NOT BOTH.
*
TPLOGIC1 MTMOD RECFORM=FIXUNB,TYPEFLE=INPUT,READ=FORWARD,WORKA=YES
```

```
* EXAMPLE
*
* DTFMT FOR FIXBLK INPUT TAPE WITHOUT LABELS.
*
*
OLDMSTR  DTFMT BLKSIZE=1980,
               DEVADDR=SYS005,
               EOFADDR=DONE,
               FILABL=NO,
               IOAREA1=TAPE1,
               IOAREA2=TAPE2,
               MODNAME=TPLOGIC3,
               RECFORM=FIXBLK,      PARAMETER CHANGE.
               RECSIZE=198,         NEW ENTRY.
               TYPEFLE=INPUT,
               IOREG=(9),           ONE OR OTHER OF
               WORKA=YES            THESE, NOT BOTH.
*
TPLOGIC3 MTMOD READ=FORWARD,RECFORM=FIXBLK,TYPEFLE=INPUT,WORKA=YES
*
*
*
* DTFMT FOR FIXBLK OUTPUT TAPE WITHOUT LABELS.
*
*
NEWMSTR  DTFMT BLKSIZE=1980,
               DEVADDR=SYS005,
               FILABL=NO,
               IOAREA1=TAPE1,
               IOAREA2=TAPE2,
               MODNAME=TPLOGIC4,
               RECFORM=FIXBLK,      PARAMETER CHANGE.
               RECSIZE=198,         NEW ENTRY.
               TYPEFLE=OUTPUT,
               IOREG=(9),           ONE OR OTHER OF
               WORKA=YES            THESE, NOT BOTH.
*
TPLOGIC4 MTMOD RECFORM=FIXBLK,TYPEFLE=OUTPUT,WORKA=YES
*
```

* EXAMPLE

PROBLEM 02. CARD TO TAPE WITH EXCEPTION FILE.

REQUIREMENTS-

1. READ INPUT CARDS FROM SYSIPT.
2. WRITE OUTPUT ON TAPE SYS004.
3. WRITE EXCEPTIONS ON TAPE SYS007.
4. SYS004 BLOCKING FACTOR IS 4.
5. SYS007 BLOCKING FACTOR IS 1.
6. IF INPUT CARD HAS A '$' IN COLUMN 4
   IT IS CONSIDERED AN EXCEPTION.
7. USE 2 I/O AREAS ON ALL FILES.
8. USE THE IOREG APPROACH ON ALL FILES.
9. READ 80 COLUMNS FROM CARD.
10. WRITE 50 COLUMNS IF EXCEPTION.
11. WRITE 60 COLUMNS FOR ALL OTHER FILES.
12. WRITE ALL THE CONTROL CARDS NECESSARY TO EXECUTE THIS JOB
    FROM THE CORE IMAGE LIBRARY.

```
* EXAMPLE [illegible].
*
* CREATE VARBLK FILE USING VARBLD APPROACH.
*
*
         REPRO
         PHASE VARBLK,S
EXAMPLE  START 0
CARD     DTFCD BLKSIZE=80,CONTROL=YES,DEVADDR=SYS001,DEVICE=2540,
               EOFADDR=LSTCD,IOAREA1=N1,RECFORM=FIXUNB,TYPEFLE=INPUT,
               MODNANE=CDLOGIC2
NMSTR    DTFMT BLKSIZE=1012,DEVADDR=SYS017,FILABL=STD,IOAREA1=OUT1,
               IOAREA2=OUT2,IOREG=(7),RECFORM=VARBLK,TYPEFLE=OUTPUT,
               VARBLD=(6),MODNAME=TPLOGIC9
CDLOGIC2 DCMOD RECFORM=FIXUNB,CONTROL=YES,TYPEFLE=INPUT,DEVICE=2540
TPLOGIC9 MTMOD RECFORM=VARBLK,TYPEFLE=OUTPUT
         DS    OH
OUT1     DC    X'00004040'
         DS    4CL252
OUT2     DC    X'00004040'
         DS    4CL252
DUMOUT   DSECT
OUTAREA  DS    CL84
HERE     CSECT
         BALR  11,0
         USING *,11
         USING DUMOUT,7
         OPEN  CARD,NMSTR
         LA    6
NEXT     GET   CARD
         LM    3,5,INIT
CMPR     CLI   0(3),C' '
         BNE   PUTREC
         BXH   3,4,CMPR
         CNTRL CARD,PS,2
         B     NEXT
PUTREC   CNTRL CARD,PS,3
         S     3,START
         STH   3,RL
         CR    6,3
         BL    TCATE
CONT     BCT   3,*+4
         EX    3,MOVE
         PUT   NMSTR
         B     NEXT
TCATE    TRUNC NMSTR
         B     CONT
LSTCD    CLOSE CARD,NMSTR
         EOJ
MOVE     MVC   OUTAREA(0),NEWREC
NEWREC   DS    OCL84
RL       DS    CL2
         DC    X'4040'
N1       DS    CL80
INIT     DC    A(N1+79)
         DC    F'-1'
         DC    A(N1-1)
START    DC    A(RL-1)
         LTORG
         END   HERE
```

```
* EXAMPLE
*
* ILLUSTRATION OF CREATING VARUNB FILE.
*
*
         REPRO
         PHASE EXAM16,S
EXAMPLE  START 0
NEWMSTR  DTFMT BLKSIZE=88,DEVADDR=SYS005,FILABL=STD,IOAREA1=OUT1,
               IOAREA2=OUT2,RECFORM=VARUNB,REWIND=UNLOAD,
               TYPEFLE=OUTPUT,WORKA=YES,MODNAME=TPLOGIC7
CARD     DTFCD BLKSIZE=80,CONTROL=YES,DEVADDR=SYSIPT,DEVICE=2540,
               EOFADDR=LSTCD,IOAREA1=IN1,RECFORM=FIXUNB,TYPEFLE=INPUT,
               WORKA=YES,MODNAME=CDLOGIC5
TPLOGIC7 MTMOD RECFORM=VARUNB,TYPEFLE=OUTPUT,WORKA=YES
CDLOGIC5 CDMOD RECFORM=FIXUNB,CONTROL=YES,TYPEFLE=INPUT,WORKA=YES,
               DEVICE=2540
OUT1     DS    H
         DC    CL2'  '
         DS    CL84
OUT2     DS    H
         DC    CL2'  '
         DS    CL84
HERE     CSECT
         BALR  11,0
         USING *,11
         B     NEXT
IN1      DS    CL80
NEWREC   DS    OLC84
RL       DS    CL2
         DC    CL2'  '
TESTAR   DS    CL80
NEXT     OPEN  CARD,NEWMSTR
GETCD    GET   CARD,TESTAR
         LM    3,5,INIT
COMP     CLI   0(3),C' '
         BNE   PUTREC
         BXH   3,4COMP
         CNTRL CARD,PS,2      SELECT CARD IF NOT 4 CHAR.  ( SEE NOTE.)
         B     GETCD
PUTREC   CNTRL CARD,PS,3
         S     3,START
         STH   3,RL
         PUT   NEWMSTR,NEWREC
         B     GETCD
LSTCD    CLOSE CARD,NEWMSTR
         EOJ
INIT     DC    A(TESTAR+79)
         DC    F'-1'
         DC    A(TESTAR+2)
START    DC    A(RL-1)
*
* NOTE- MINIMUM SIZE TAPE RECORD PERMISSABLE USING IOCS IS 12 CHAR.
         LTORG
         END   HERE
```

```
* EXAMPLE .
*
* ILLUSTRATION USING UNDEF APPROACH WITH TAPE FILES.
* CARD TO TAPE  --  THEN TAPE TO PRINT
*
         REPRO
         PHASE UNDEF,S
EXAMPLE  START 0
CARD     DTFCD BLSKIZE=80,CONTROL=YES,DEVADDR=SYS001,DEVICE=2540,
               EOFADDR=LSTCD,IOAREA1=N1,RECFORM=FIXUNB,TYPEFLE=INPUT,
               MODNAME=CDLOGIC2
RECOUT   DTFMT BLKSIZE=80,DEVADDR=SYS011,FILABL=STD,IOAREA1=N1,
               RECFORM=UNDEF,RECSIZE=(3),TYPEFLE=OUTPUT,
               MODNAME=TPLOGICA
RECIN    DTFMT BLKSIZE=80,DEVADDR=SYS011,EOFADDR=E,FILABL=STD,
               IOAREA1=N1,RECFORM=UNDEF,RECSIZE=(3),TYPEFLE=INPUT,
               MODNAME=TPLOGICB
LINE     DTFPR BLKSIZE=80,DEVADDR=SYS002.DEVICE=1403.IOAREA1=N1,
               PRINTOV=YES,RECFORM=UNDEF,RECSIZE=(3),MODNAME=PRLOGIC6
CDLOGIC2 CDMOD RECFORM=FIXUNB,CONTROL=YES,TYPEFLE=INPUT,DEVICE=2540
TPLOGICA MTMOD RECFORM=UNDEF,TYPEFLE=OUTPUT
TPLOGICB MTMOD READ=FORWARD,RECFORM=UNDEF,TYPEFLE=INPUT
PRLOGIC6 PRMOD RECFORM=UNDEF,PRINTOV=YES
HERE     CSECT
         BALR  11,0
         USING *,11
         OPEN  CARD,RECOUT
GET1     GET   CARD
         LM    3,5,INIT
COMP     CLI   0(3),C' '          SCAN FOR FIRST SIGNIFICANT CHARACTER.
         BNE   WRITE
         BXH   3,4,COMP
         CNTRL CARD,PS,2          NOT 12 CHAR. IN RECORD.
         B     GET1
WRITE    CNTRL CARD,PS,3
         S     3,START
         PUT   RECOUT
         B     GET1
LSTCD    CLOSE CARD,RECOUT
         OPEN  RECIN,LINE
GET2     GET   RECIN
         PUT   LINE
         PRTOV LINE,12
         B     GET2
E        CLOSE RECIN,LINE
         EOJ
N1       DS    CL80
INIT     DC    A(N1+79)
         DC    F'-1'
         DC    A(N1+11)
START    DC    A(N1-1)
         LTORG
         END   HERE
```

* EXAMPLE

PROBLEM 03. MERGE MASTER FILE WITH EXCEPTION FILE.

REQUIREMENTS-

1. MASTER FILE ON SYS009.
2. EXCEPTION FILE ON SYS005.
3. NEW MASTER ON SYS006.
4. MASTER BLOCKED 6. 60 CHARACTER LOGICAL RECORDS.
5. EXCEPTION FILE UNBLOCKED. 50 CHARACTER LOGICAL RECORD.
6. NEW MASTER BLOCKED 10. 80 CHARACTER LOGICAL RECORDS.
7. PAD EXCEPTION AND MASTER RECORDS WITH BLANKS TO OBTAIN NEW MASTER RECORD LENGTH OF 80.
8. CONTROL FIELD IS IN COLUMNS 6-10 OF ALL RECORDS.
9. USE WORKAREA APPROACH.
10. USE TWO AREAS ON ALL FILES.
11. ALL FILES HAVE STANDARD LABELS.
12. CONTROL CARDS TO EXECUTE PROGRAM FROM OBJECT DECK.

```
* EXAMPLE
*
* SOLUTION PROBLEM 03.
*
*
         REPRO
         PHASE PROB3,S
EXAMPLE  START 0
OLDMST   DTFMT BLKSIZE=360,DEVADDR=SYS009,EOFADDR=LSTRCD,FILABL=STD,
               IOAREA1=N1,IOAREA2=N2,RECFORM=FIXBLK,RECSIZE=60,
               TYPEFLE=INPUT,WORKA=YES,MODNAME=TPLOGIC3
EXCEPT   DTFMT BLKSIZE=50,DEVADDR=SYS005,EOFADDR=LSTEXC,FILABL=STD,
               IOAREA1=N3,IOAREA2=N4,RECFORM=FIXUNB,TYPEFLE=INPUT,
               WORKA=YES,MODNAME=PTLOGIC3
NEWOUT   DTFMT BLKSIZE=800,DEVADDR-SYS006,FILABL=STD,IOAREA1=OUT,
               IOAREA2=OUT2,RECFORM=FIXBLK,RECSIZE=80,TYPEFLE=OUTPUT,
               WORKA=YES,MODNAME=TPLOGIC3
TPLOGIC3 MTMOD READ=FORWARD,RECFORM=FIXBLK,TYPEFLE=INPUT,WORKA=YES
N1       DS    6CL60
N2       DS    6CL60
N3       DS    CL50
N4       DS    CL50
OUT      DS    10CL80
OUT2     DS    10CL80
BEGIN    CSECT
         BALR  11,0
         USING *,11
         MVC   WORK1,(160),BLANK
         OPEN  OLDMST,EXCEPT,NEWOUT
         GET   EXCEPT,WORK1
GET1     GET   OLDMST,WORK2
COMP     CLC   EXC,OLD
         BNH   PUT1
         PUT   NEWOUT,WORK2
         B     GET1
PUT1     PUT   NEWOUT,WORK1
         GET   EXCEPT,WORK1
         B     COMP
LSTRCD   OI    SW1,X'FF'
         CLI   SW2,X'FF'
         BE    ENDRT
         MVC   OLD,=X'FFFFFFFFFF'
         B     COMP
LSTEXC   OI    SW2,X'FF'
         CLI   SW1,X'FF'
         BE    ENDRT
         MVC   EXC,=X'FFFFFFFFFF'
         B     COMP
ENDRT    CLOSE OLDMST,EXCEPT,NEWOUT
         EOJ
BLANK    DC    C' '
WORK1    DS    CL5
EXC      DS    CL5
         DS    CL70
WORK2    DS    CL5
OLD      DS    CL5
         DS    CL70
SW1      DC    C' '
SW2      DC    C' '
         LTORG
         END   BEGIN
```

* EXAMPLE

PROBLEM 04. CREATE VARBLK FILE.

1. INPUT IS CARD READER (SYSIPT).
2. DATA IN CARD IS LEFT JUSTIFIED.
3. OUTPUT FILE IS SYS005. STANDARD LABELS.

REQUIREMENTS-

4. CREATE ONE VARIABLE LENGTH RECORD FOR EACH CARD OF INPUT DATA.
5. SELECT INPUT DATA CARDS INTO POCKET 3.
6. SKIP BLANK INPUT CARDS AND SELECT THEM INTO POCKET 2.
7. OUTPUT FILE RECFORM IS VARBLK.
8. OUTPUT I/O AREA IS LARGE ENOUGH TO HOLD 12 MAXIMUM LENGTH (80 CHAR) INPUT RECORDS.
9. USE WORKAREA APPROACH TO CREATE OUTPUT FILE.
10. USE TWO I/O AREAS FOR THE OUTPUT FILE.
11. CONTROL CARDS TO EXECUTE PROGRAM FROM RELOCATABLE LIBRARY.

```
* EXAMPLE
*
* SOLUTION PROBLEM 4.
*
*

         REPRO
         PHASE PROB4,S
EXAMPLE  START 0
NEWMSTR  DTFMT BLKSIZE=1012,DEVADDR=SYS005,FILABL=STD,IOAREA1,=OUT1,
               IOAREA2=OUT2,RECFORM=VARBLK,REWIND=UNLOAD,
               TYPEFLE=OUTPUT,WORKA=YES MODNAME=TPLOGIC8
CARD     DTFCD BLKSIZE=80,CONTROL=YES,DEVADDR=SYSIPT,DEVICE=2540.
               EOFADDR=LSTCD,IOAREA1=TESTAR,RECFORM=FIXUNB,
               TYPEFLE=INPUT,MODNAME=CDLOGIC5
TPLOGIC8 MTMOD RECFORM=VARBLK,TYPEFLE=OUTPUT,WORKA=YES
CDLOGIC5 CDMOD RECFORM=FIXUNB,CONTROL=YES,TYPEFLE=INPUT,
               DEVICE=2540
OUT1     DS    H
         DC    CL2'  '
         DS    12CL84
OUT2     DS    H
         DC    CL2'  '
         DS    12CL84
ENTER    CSECT
         BALR  11,0
         USING *,11
         OPEN  CARD,NEWMSTR
GET1     GET   CARD
         LM    3,8,INIT
COMP     CLI   0(3),C' '
         BNE   PUTREC
         BXH   3,4,COMP
         CNTRL CARD,PS,2
         B     GET1
PUTREC   CNTRL CARD,PS,3
         S     3,START
         STH   3,RL
         PUT   NEWMSTR,NEWREC
         B     GET1
LSTCD    CLOSE CARD,NEWMSTR
         EOJ
NEWREC   DS    0CL84
RL       DS    CL2
         DC    X'4040'
TESTAR   DS    CL80
INIT     DC    A(TESTAR+79)
         DC    F'-1'
         DC    A(TESTAR-1)
START    DC    A(RL-1)
         LTORG
         END   ENTER
```

ASSEMBLY LANGUAGE CODING
INPUT/OUTPUT CONTROL SYSTEMS
FOR
DIRECT ACCESS FILES

```
* EXAMPLE
*
* DTFDA FOR CREATING FILE
*
*
FILENAME DTFDA AFTER=YES,
               BLKSIZE=N,   (SEE NOTE 2.)
               DEVICE=2311 OR 2321,
               ERRBYTE=NAME,
               IOAREA1=NAME, (SAME SIZE AS BLKSIZE)
               KEYLEN=N,    (USE ONLY IF KEY IS TO BE WRITTEN.)
               MODNAME=DAMOD2,
               RECFORM=FIXUNB, (OR UNDEF)
               SEEKADR=NAME,  (SEE NOTE 1.)
               TYPEFLE=OUTPUT,
               VERIFY=YES    (OPTIONAL)
DAMOD2   DAMOD AFTER=YES,RECFORM=FIXUNB
```

NOTE 1. THE FIELD STORAGE INDICATED BY THE NAME IN THE SEEKADR ENTRY IS AN EIGHT BYTE FIELD OF THE FORMAT 'MBBCCHHR'.

NOTE 2. BLKSIZE EQUAL TO COUNT + KEY + DATA IF KEYLEN ENTRY IS USED. BLKSIZE EQUAL TO COUNT + DATA IF KEYLEN NOT USED.

```
* EXAMPLE
*
* ILLUSTRATION OF LOADING A RANDOM FILE USING 'AFTER' APPROACH.
*
*       ASSUME-
*
*         1. RECORDS COME FROM TAPE. (SYS019)
*         2. KEY IS CONTROL FIELD IN POS 1-10 IN EACH TAPE RECORD.
*         3. TAPE RECORDS ARE 110 CHARACTERS LONG.
*         4. A RANDOMIZING ROUTINE IS ALREADY WRITTEN TO TAKE A BINARY
*            VALUE IN GEN. REG. 2 AND RETURN CYL NO. IN GEN. REG. 2 AND
*            HEAD NO. IN GEN. REG. 3. THE NAME OF THE ROUTINE IS 'RDMIZ'
*            AND IT USES GEN. REG. 14 FOR RETURN LINKAGE.
*
         REPRO
         PHASE LOADRDM,S
EXAMPLE  START 0
         EXTRN RDMIZ
TAPE     DTFMT BLKSIZE=110,DEVADDR=SYS019,EOFADDR=EOJRT,FILABL=STD,
               IOAREA1=N1,RECFORM=FIXUNB,REWIND=UNLOAD,TYPEFLE=INPUT,
               MODNAME=TPLOGIC
DISK     DTFDA AFTER=YES,BLKSIZE=118,DEVICE=2311,
               ERRBYTE=FLAGS,IOAREA1=OUT1,RECFORM=FIXUNB,
               TYPEFLE=OUTPUT,SEEKADR=FIND,VERIFY=YES,KEYLEN=10,
               MODNAME=DSKLOGIC
TPLOGIC  MTMOD READ=FORWARD,RECFORM=FIXUNB,TYPEFLE=INPUT
DSKLOGIC DAMOD AFTER=YES,RECFORM=FIXUNB
*
HERE     CSECT
         BALR  11,0
         USING *,11
         OPEN  TAPE,DISK
NEXT     GET   TAPE
         PACK  PKAR,KEY
         CVB   2,PKAR
         L     15,ADDR1
         BALR  14,15
         STH   2,FIND+3
         STH   3,FIND+5
         WRITE DISK,AFTER
         WAITF DISK
         CLC   FLAGS,=X'0000'
         BE    NEXT
            .
            .  ERROR ROUTINE
            .
EOJRT    CLOSE TAPE,DISK
         EOJ
PKAR     DS    D
ADDR1    DC    A(RDMIZ)
FLAGS    DC    X'0000'
         DC    X'00'
FIND     DC    X'000000'
         DC    CL5
OUT1     DS    OCL118
         DS    CL8
N1       DS    OCL110
KEY      DS    CL10
DATA     DS    CL100
         LTORG
         END   HERE
```

```
* EXAMPLE
*
* ILLUSTRATION OF A DTFDA FOR RANDOM INQUIRY BY ID.
*
*
FILENAME DTFDA BLSKIZE=N,
               CONTROL=YES,   (OPTIONAL)
               DEVICE=2311 OR 2321,
               ERRBYTE=NAME,
               IOAREA1=NAME,
               KEYLEN=N,      (USE ONLY IF KEY DESIRED)
               MODNAME=DSLOG,
               READID=YES,
               RECFORM=FIXUNB, (ASSUMED IF OMITTED)
               SEEKADR=NAME,
               TYPEFEL=INPUT  (INDICATES TYPE OF LABEL CHECKING)

DSLOG    DAMOD RECFORM=FIXUNB
```

* EXAMPLE
*
* PROBLEM 07. RANDOM RETRIEVAL BY ID APPROACH.

1. TAPE INPUT. FIXUNB. RECORD LENGTH 56. SYS027.

2. DISK FILE START LOCATION = 1200 (CYL 120-TRACK 0)
   FIXUNB RECORDS. KEY LENGTH 10. DATA LENGTH 260.

3. 7560 ACCOUNTS - 10 ACCOUNTS/TRACK.
   ACCOUNT NUMBER RANGE IS 10,000 - 17,560.
   ACCOUNT NUMBER IS POSITIONS 1-5 OF TAPE RECORD.
   ACCOUNT NUMBER IS ZONED DECIMAL FORMAT.

4. ACCOUNT NUMBER DETERMINES ID.

   FORMULA TO COMPUTE ID IS-

   (ACCT NO. - 10,000)/10+ 1200 = CCHH

   REMAINDER OF (ACCT NO. - 10,000)/10+1 =R

5. PRINT TAPE RECORD FOLLOWED BY DISK RECORD RETRIEVED AS A RESULT OF TAPE RECORD ACCOUNT NUMBER. PRINTER SYS014.

6. JOB CONTROL CARDS TO ASSEMBLE AND EXECUTE.

```
* EXAMPLE
*
* ILLUSTRATION OF A DTFDA FOR RANDOM INQUIRY BY KEY.
*
*
FILNAME  DTFDA BLKSIZE=N,
               DEVICE=2311 OR 2321,
               ERRBYTE=NAME,
               IOAREA1=NAME,
               KEYARG=NAME,   (PLACE KEY OF RECORD DESIRED HERE)
               KEYLEN=N,      (MANDATORY FOR RETRIEVAL BY KEY)
               MODNAME=DSKLOGIC,
               READKEY=YES,
               RECFORM=FIXUNB,
               SEEKADR=NAME,
               TYPEFLE=INPUT,
               SRCHM=YES
*
DSKLOGIC DAMOD RECFORM=FIXUNB
```

```
* EXAMPLE
*
* SOLUTION PROBLEM 07.
*
*
         REPRO
         PHASE RDMRTV,S
EXAMPLE  START 0
TAPE     DTFMT BLKSIZE=56,DEVADDR=SYS027,EOFADDR=END,FILABL=STD,
               IOAREA1=N1,RECFORM=FIXUNB,REWIND=UNLOAD,TYPEFLE=INPUT,
               MODNAME=TPLOGIC
LINE     DTFPR BLKSIZE=132,DEVADDR=SYS014,DEVICE=1403,IOAREA1=OUT1,
               PRINTOV=YES,RECFORM=UNDEF,RECSIZE=(2),MODNAME=PRLOGIC
DISK     DTFDA BLKSIZE=270,DEVICE=2311,ERRBYTE=FLAGS,IOAREA1=N2,
               KEYLEN=10,READID=YES,RECFORM=FIXUNB,SEEKADR=MOD,
               TYPEFLE=INPUT,MODNAME=RNDRTVE
*
PRLOGIC  PRMOD PRINTOV=YES,RECFORM=UNDEF
*
*
TPLOGIC  MTMOD READ=FORWARD,RECFORM=FIXUNB,TYPEFLE=INPUT

RNDRTVE  DAMOD RECFORM=FIXUNB
HERE     CSECT
         BALR  11,0
         USING *,11
         OPEN  LINE,TAPE,DISK
G1       GET   TAPE
         PACK  PKAR,CTL
         CVB   4,PKAR                BINARY CT. NO.
         L     6,K1                  LOAD SUBTRAHEND.
         SR    4,6                   COMPUTE DIVIDEND.
         LA    6,10                  CREATE DIVISOR.
         SRDA  4,32                  CLEAR AND INIT. FOR DIVIDE.
         DR    4,6                   DIVIDE BY 10.
         LA    7,1                   CREATE CONSTANT 1.
         AR    4,7                   COMPUTE RECORD NUMBER.
         STC   4,REC                 STORE RECORD NUMBER IN ID FIELD.
         SR    4,4                   CLEAR REGISTER 4.
         DR.   4,6                   DIVIDE BY TRACKS/CYLINDER.
         STH   4,HH                  STORE HH VALUE IN ID FIELD.
         LA    6,120                 CREATE CONSTANT 120.
         AR    6,5                   COMPUTE CYL. NO. FROM STARTING ADDRESS.
         STH   6,CC                  STORE CC VALUE IN ID FIELD.
         LA    2,56
         MVC   OUT1(56),N1
         PUT   LINE
         READ  DISK,ID
         WAITF DISK
         CLC   FLAGS,=X'0000'
         BNE   ERROR
         MVC   OUT1,OUT2
         LA    2,132
         PUT   LINE
         MVC   OUT1,OUT3
         PUT   LINE
         MVC   OUT1(6),OUT4
         LA    2,6
         PUT   LINE
         PRTOV LINE,12
         B     G1
ERROR
```

```
                .      ERROR ROUTINE
                .
END       CLOSE  LINE,TAPE,DISK
          EOJ
K1        DC     F'10000'
FLAGS     DC     X'0000'
          DC     X'00'
MOD       DC     X'00'
BIN       DC     X'0000'
CC        DS     CL2
HH        DC     CL2
REC       DS     CL1
CTL       DS     OCL5
N1        DS     CL56
OUT1      DS     CL132
OUT2      DS     CL132
OUT3      DS     CL132
OUT4      DS     CL6
PKAR      DS     D
N2        EQU    OUT2
          LTORG
          END    HERE
```

ASSEMBLY LANGUAGE CODING
INPUT/OUTPUT CONTROL SYSTEMS
FOR
INDEX SEQUENTIAL FILES

```
* EXAMPLE [illegible]
*
* LOAD INDEX SEQUENTIAL FILE (FROM A SORTED TAPE FILE). THIS EXAMPLE
        ASSUMES THE KEY FIELD IS POS. 11-15 IN EACH LOGICAL RECORD.
*
        REPRO
        PHASE LOADIS,S
EXAMPLE START 0
TAPE    DTFMT BLKSIZE=6660,DEVADDR=SYS015,EOFADDR=ENDTP,FILABL=STD,
              IOAREA1=N1,IOAREA2=N2,RECFORM=FIXBLK,RECSIZE=111,
              TYPEFLE=INPUT,WORKA=YES,MODNAME=TPLOG
MESG    DTFCN BLKSIZE=16,DEVADDR=SYSLOG,IOAREA1=OUT,RECFORM=FIXUNB,
              TYPEFLE=OUTPUT,WORKA=YES
DISK    DTFIS CYLOFL=2,DEVICE=2311,DSKXTNT=3,IOAREAL=OUT1,IOROUT=LOAD,
              KEYLEN=5,KEYLOC=11,NRECDS=10,RECFORM=FIXBLK,RECSIZE=111,
              VERIFY=YES,WORKL=DATAR,MODNAME=DSKL,HINDEX=2311
TPLOG   MTMOD READ=FORWARD,RECFORM=FIXBLK,TYPEFLE=INPUT,WORKA=YES
*
DSKL    ISMOD IOROUT=LOAD
*
N1      DS    60CL111
N2      DS    60CL111
OUT     DS    CL16
OUT1    DS    CL8
        DS    CL5
        DS    10CL111
THERE   CSECT
        BALR  11,0
        USING *,11
        OPEN  TAPE,DISK
        SETFL DISK
        BAL   6,ROUT
REFILL  GET   TAPE,DATAR
        WRITE DISK,NEWKEY
        BAL   6,ROUT
        B     REFILL
ROUT    L     4,=A(DISKC)
        TM    0(4),X'FE'
        BM    ERR
        BR    6
ERR     TM    0(4),X'80"
        BO    DSKERR
        TM    0(4),X'40"
        BO    WLREC
        TM    0(4),X'20'
        BO    DATAOF
        TM    0(4),X'10'
        BO    CYLOF
        TM    0(4),X'04'
        BO    DUPREC
        TM    0(4),X'02'
        BO    SEQERR
CYLOF   PUT   MESG,ONE
        B     EOJ
DSKERR  PUT   MESG,TWO
        B     EOJ
DATAOF  PUT   MESG,THREE
        B     EOJ
DUPREC  PUT   MESG,FOUR
        B     EOJ
SEQERR  PUT   MESG,FIVE
        B     EOJ
```

```
WLREC          PUT   MESG,SIX
               B     EOJ
ENDTP          ENDFL DISK
               BAL   6,ROUT
               CLOSE TAPE,DISK
EOJ            EOJ
DATAR          DS    CL111
ONE            DC    C'CYL INDEX FULL.  '
TWO            DC    C'DISK ERROR.      '
THREE          DC    C'DATA AREA OVFLO.'
FOUR           DC    C'DUPLICATE KEY.   '
FIVE           DC    C'SEQ CHECK ERROR.'
SIX            DC    C'WRONG LNGTH REC.'
               LTORG
               END   THERE
```

```
* EXAMPLE [illegible].
*
* PROBLEM 05. ADD TO INDEX SEQUENTIAL FILE.
*
*
           1.  READ FILE ADDITIONS FROM SYSIPT.
           2.  FILE NAME IS 'CARD'.
           3.  KEY IS IN POSITIONS 1-10.
           4.  RECORDS ARE 80 POSITIONS LONG.

           5.  DISK FILE IS BLOCKED 10.
           6.  FILENAME IS 'DISK'.
           7.  TWO OVERFLOW TRACKS PER CYLINDER.
           8.  3 EXTENT JOB CARDS DESCRIBE THE FILE.

           9.  PRINTER IS SYS008.
           10. PRINTER FILENAME IS 'LINE'.
           11. PRINT THE KEY NO. FOR EACH ADDITION.
           12. CHECK FOR PAGE OVERFLOW.
           13. PRINT PAGE NO. AND JOB NO. AT TOP OF EACH PAGE.
           14. OVERFLOW PUNCH IS IN CHANNEL 12 OF CARRIAGE TAPE.
           15. SKIP TO CHANNEL TWO AFTER PRINTING HEADING LINE.

           16. NOTIFY OPERATOR ON ANY ERROR CONDITION AND TERMINATE JOB.
```

```
* EXAMPLE 7 .
*
* SOLUTION PROBLEM 5.
*
*
         REPRO
         PHASE PROB5,S
EXAMPLE  START 0
CARD     DTFCD BLKSIZE=80,DEVADDR=SYSIPT,DEVICE=2540.EOFADDR=LSTCD,
               IOAREA1=A1,RECFORM=FIXUNB,TYPEFLE=INPUT,MODNAME=CDLOGIC
LINE     DTFPR BLKSIZE=25,CONTROL=YES,DEVADDR=SYS008,DEVICE=1403,
               IOAREA1=L1,RECFORM=UNDEF,RECSIZE=(2),WORKA=YES,
               PRINTOV=YES,MODNAME=PRLOGIC
DISK     DTFIS CYCLOFL=2,DSKXTNT=3,DEVICE=2311,IOAREAL=D1,IOROUT=ADD,
               KEYLEN=10,KEYLOC=1,MODNAME=DSKA,NRECDS=10,HINDEX=2311,
               RECFORM=FIXBLK,RECSIZE=80,VERIFY=YES,WORKL=A1
*
CDLOGIC  CDMOD DEVICE=2540,RECFORM=FIXUNB,TYPEFLE=INPUT
*
PRLOGIC  PRMOD CONTROL=YES,PRINTOV=YES,RECFORM=UNDEF,WORKA=YES
*
DSKA     ISMOD IOROUT=ADD,RECFORM=FIXBLK
HERE     CSECT
         BALR  11,0
         USING *,11
         OPEN  CARD,LINE,DISK
         BAL   14,HDRT
GET1     GET   CARD
         PUT   LINE,A1
         WRITE DISK,NEWKEY
         WAITF DISK
         BAL   6,ROUT
         CNTRL LINE,SP,1
         PRTOV LINE,12,HDRT
         B     GET1
HDRT     ST    14,HOLD
         AP    PGNO,ONE
         UNPK  HDLN,PGNO
         OI    HDLN+1,X'F0'
         LA    2,24
         CNTRL LINE,SK,1
         PUT   LINE,PGHD
         CNTRL LINE,SK,2
         LA    2,10
         L     14,HOLD
         BR    14
ROUT     L     4,=A(DISKC)
         TM    0(4),X'FE'
         BM    ERROR
         BR    6
FULL     LA    2,20
         CNTRL LINE,SK,1
         PUT   LINE,MSG1
         CNTRL LINE,SK,1
         B     EOJ
ERR      LA    2,25
         CNTRL LINE,SK,1
         PUT   LINE,MSG2
         CNTRL LINE,SK,1
         B     EOJ
DUP      LA    2,14
         CNTRL LINE,SK,1
```

```
            PUT   LINE,MSG3
            CNTRL LINE,SK,1
            B     EOJ
WLR         LA    2,20
            CNTRL LINE,SK,1
            PUT   LINE,MSG4
            CNTRL LINE,SK,1
            B     EOJ
ERROR       TM    0(4),X'80"
            BO    ERR
            TM    0(4),X'40'
            BO    WLR
            TM    0(4),X'04'
            BO    DUP
            TM    0(4),X'02'
            BO    FULL
LSTCD       LA    2,11
            CNTRL LINE,SK,1
            PUT   LINE,MSG5
            CNTRL LINE,SK,1
            CLOSE CARD,LINE,DISK
EOJ         EOJ
MSG1        DC    C'OVERFLOW TRACK '
            DC    C'FULL.'
MSG2        DC    C'UNCORRECTABLE '
            DC    C'DISK ERROR.'
MSG3        DC    C'DUPLICATE KEY.'
MSG4        DC    C'WRONG LENGTH '
            DC    C'RECORD.'
MSG5        DC    C'END OF JOB.'
HOLD        DS    F
A1          DS    CL80
L1          DS    CL25
D1          DS    CL18
            DS    10CL80
ONE         DC    X'1C'
PGNO        DC    X'000C'
PGHD        DC    C'PAGE NO. '
HDLN        DC    C'  '
            DC    C'JOB NO. 46.'
W1          DS    CL80
            LTORG
            END   HERE
```

```
* EXAMPLE
*
* ILLUSTRATION OF A DTFIS MACRO FOR RETRIEVAL FROM AN IS FILE.
*
*
FILENAME DTFIS DSKXTNT=N,
               DEVICE=2311,  OR 2321
               HINDEX=2311,  OR 2321
               IOAREAS=NAME,
               IOROUT=RETRVE,
               KEYARG=AREANAME,  USE ONLY IF INITIATED BY KEY OR GKEY,
               KEYLEN=N,
               KEYLOC=N,
               MODNAME=NAME,
               NRECDS=N,
               RECFORM=FIXUNB OR FIXBLK,
               RECSIZE=N,
               TYPEFLE=SEQNTL,
               IOREG=(N)  (USE 'IOREG' OR 'WORKS' ENTRY - NOT BOTH.
               WORKS=YES  EITHER 'IOREG' OR 'WORKS' SHOULD BE USED
*                         IN AN INDEX SEQUENTIAL FILE EVEN THOUGH THEY
*                         ARE NOT MANDATORY. THIS IS BECAUSE IT IS
*                         NEVER KNOWN WHETHER THE RECORD COMES FROM
*                         A DATA AREA OR AN OVERFLOW AREA. THEREFORE,
*                         THE STARTING POSITION OF THE INPUT DATA IN
*                         I/O AREA MAY VARY BY 10 POSITIONS BECAUSE
*                         OF THE SEQUENCE LINK FIELD IN AN OVERFLOW
*                         RECORD. BY USING 'IOREG' OR 'WORKS' IOCS
*                         WILL SUPPLY ONLY THE DATA AND IGNORE THE
*                         SEQUENCE LINK FIELD.)
*
*
*
*
NAME     ISMOD IOROUT=RETRVE,
               TYPEFLE=SEQNTL
```

```
* EXAMPLE
*
* PROBLEM 06. SEQUENTIAL RETRIEVAL - FILE REORGANIZATION - ADD - DELETE.
*
*
            1.  RETRIEVE RECORDS FROM AN INDEX SEQUENTIAL MASTER FILE
                BEGINNING WITH KEY NO. AS INDICATED IN A LEAD CARD.

            2.  RELOAD ALL RECORDS EXCEPT THOSE WITH A 'D' IN POSITION 75
                ON A NEW INDEX SEQUENTIAL FILE.

            3.  PRINT ALL DELETED RECORDS, PRINTER IS SYSLST.

            4.  LOGICAL RECORDS ARE 80 POSITIONS LONG. KEY IS IN POS. 1-5.

            5.  DISK FILES ARE BLOCKED 10. THERE ARE TWO OVERFLOW TRACKS
                PER CYLINDER.

            6.  THERE ARE THREE EXTENT CARDS FOR EACH FILE.

            7.  PRINT AN ERROR MESSAGE AND TERMINATE JOB ON ANY DISK FILE
                ERROR CONDITION.

            8.  MERGE ADDITIONS READ FROM CARD READER SYSIPT. LEAD CARD
                INDICATES STARTING KEY NO. IN POSITION 1-5.
```

```
* EXAMPLE
*
* SOLUTION PROBLEM 06.
*
*
         REPRO
         PHASE PROB6,S
EXAMPLE  START 0
DISKI    DTFIS DSKXTNT=3,DEVICE=2311,IOAREAS=A1,IOROUT=RETRVE,
               KEYARG=KEY,KEYLEN=5,KEYLOC=1,MODNAME=DSKINS,NRECDS=10,
               RECFORM=FIXBLK,RECSIZE=80,TYPEFLE=SEQNTL,WORKS=YES,
               HINDEX=2311
DISKO    DTFIS CYLOFL=2,DSKXTNT=3,DEVICE=2311,IOAREAL=O1,IOROUT=LOAD,
               KEYLEN=5,KEYLOC=1,MODNAME=DSKOUTL.NRECDS=10,
               RECFORM=FIXBLK,RECSIZE=80,VERIFY=YES,WORKL=WRKL,
               HINDEX=2311
CARD     DTFCD BLKSIZE=80,DEVADDR=SYSIPT,DEVICE=2540,EOFADDR=LSTCD,
               IOAREA1=A2,RECFORM=FIXUNB,SSELECT=2,TYPEFLE=INPUT,
               MODNAME=CDLOGIC
LINE     DTFPR BLKSIZE=80,CONTROL=YES,DEVADDR=SYSLST,DEVICE=1403,
               IOAREA1=WRK1,PRINTOV=YES,RECFORM=FIXUNB,MODNAME=PRLOGIC
*
DSKINS   ISMOD IOROUT=RETRVE,TYPEFLE=SEQNTL
*
DSKOUTL  ISMOD IOROUT=LOAD
*
CDLOGIC  CDMOD DEVICE=2540,RECFORM=FIXUNB,TYPEFLE=INPUT
*
PRLOGIC  PRMOD CONTROL=YES,PRINTOV=YES,RECFORM=FIXUNB
*
HERE     CSECT
         BALR  11,0
         USING *,11
         OPEN  DISKI,DISKO,CARD,LINE
         GET   CARD
         L     8,=A(DISIC)
         L     9,=A(DISKOC)
         MVC   KEY,A2
         SETL  DISKI,KEY
         BAL   6,ROUTIN
         SETFL DISKO
         BAL   7,ROUTOUT
         GET   CARD
GET1     GET   DISKI,WRK1
         BAL   6,ROUTIN
         CLI   WRK1+74,C'D'
         BE    DELETE
COMP     CLC   A2(5),WRK1
         BNH   PUTCD
         MVC   WRKL,WRK1
         WRITE DISKO,NEWKEY
         BAL   7,ROUTOUT
         B     GET1
PUTCD    MVC   WRKL,A2
         WRITE DISKO,NEWKEY
         BAL   7,ROUTOUT
         GET   CARD
         B     COMP
DELETE   PUT   LINE
         CNTRL LINE,SP,1
         PRTOV LINE,12
         B     GET1
```

```
EX1      MVC   WRK1,BLANKS
         MVC   WRK1(15),MSG1
         CNTRL LINE,SK,1
         PUT   LINE
         CNTRL LINE,SK,1
         B     EOJ
EX2      MVC   WRK1,BLANKS
         MVC   WRK1(11),MSG2
         CNTRL LINE,SK,1
         PUT   LINE
         CNTRL LINE,SK,1
         B     EOJ
EX3      MVC   WRK1,BLANKS
         MVC   WRK1(15),MSG3
         CNTRL LINE,SK,1
         PUT   LINE
         CNTRL LINE,SK,1
         B     EOJ
EX4      MVC   WRK1,BLANKS
         MVC   WRK1(32),MSG4
         CNTRL LINE,SK,1
         PUT   LINE
         CNTRL LINE,SK,1
         B     EOJ
EX5      MVC   WRK1,BLANKS
         MVC   WRK1(16),MSG5
         CNTRL LINE,SK,1
         PUT   LINE
         CNTRL LINE,SK,1
         B     EOJ
REX      MVC   WRK1,BLANKS
         MVC   WRK1(15),=C'NO RECORD FOUND'
         CNTRL LINE,SK,1
         PUT   LINE
         CNTRL LINE,SK,1
         B     EOJ
EOF      MVI   SW1,X'FF'
         CLI   SW2,X'FF'
         BW    FINISH
         MVC   WRK1(5),=X'FFFFFFFFFF'
         B     COMP
LSTCD    MVI   SW2,X'FF'
         CLI   SW1,X'FF'
         BE    FINISH
         MVC   A2(5),=X'FFFFFFFFFF'
         B     COMP
ROUTIN   TM    0(8),X'FE'
         BM    ERROR1
         BR    6
ROUTINT  TM    0(9),X'FF'
         BM    ERROR2
         BR    7
ERROR1   TM    0(8),X'80'
         BO    EX2
         TM    0(8),X'40'
         BO    EX5
         TM    0(8),X'20'
         BO    EOF
         TM    0(8),X'10'
         BO    REX
        MVC    WRK1,BLANKS
        MVC    WRK1(25),=C'END OF ERROR1 && NO BRANCH'
        CNTRL LINE,SK,1
        PUT   LINE
        CNTRL LINE,SK,1
        DUMP
ERROR2  TM    0(9),X'80'
        BO    EX2
        TM    0(9),X'40'
        BO    EX5
        TM    0(9),X'20'
        BO    EX3
        TM    0(9),X'10'
        BO    EX1
        TM    0(9),X'04'
        BO    EX4
        TM    0(9),X'02'
        BO    EX4
        MVC   WRK1,BLANKS
        MVC   WRK1(25),=C'END OF ERROR2 && NO BRANCH'
        CNTRL LINE,SK,1
        PUT   LINE
        CNTRL LINE,SK,1
        DUMP
FINISH  ESETL DISKI
        BAL   6,ROUTIN
        ENDFL DISKO
        BAL   7,ROUTOUT
        CLOSE CARD,LINE,DISKI,DISKO
EOJ     EOJ
A1      DS    10 CL80
O1      DS    CL13
        DS    10CL80
WRKL    DS    CL80
A2      DS    CL80
BLANKS  DC    C' '
WRKL    DS    CL80
SW1     DC    X'00'
SW2     DC    X'00'
MSG1    DC    C'CYL INDEX FULL.'
MSG2    DC    C'DISK ERROR.'
MSG3    DC    C'DATA AREA FULL.'
MSG4    DC    C'SEQUENCE ERROR OR DUPLICATE KEY'
MSG5    DC    C'WRONG LNGTH REC.'
KEY     DS    CL5
        LTORG
        END   HERE
```

ASSEMBLY LANGUAGE CODING

GENERAL MACHINE PROBLEM W/SOLUTIONS

MACHINE PROBLEM 9A

1. Copy Tape (SYS006) deleting all tape records with a control number matching the control number indicated in deletion cards read from SYSIPT.
2. The control number is in positions 73-80 of each tape record. Logical record length is 80 positions.
3. Input tape (SYS006) is blocked 6.
4. Input tape has standard labels.
5. Deletion cards (read from SYSIPT) have control number in columns 15 through 22.
6. Output tape (SYS004) is blocked 12.
7. Output tape is not labeled.
8. Punch a card image of all tape records deleted.
9. Verify output tape (SYS004) by printing its contents on SYSLST after it has been created. Print 72 characters.
10. Follow the logic indicated in the block diagram on the attached page to assure compatibility of this problem with future problems.

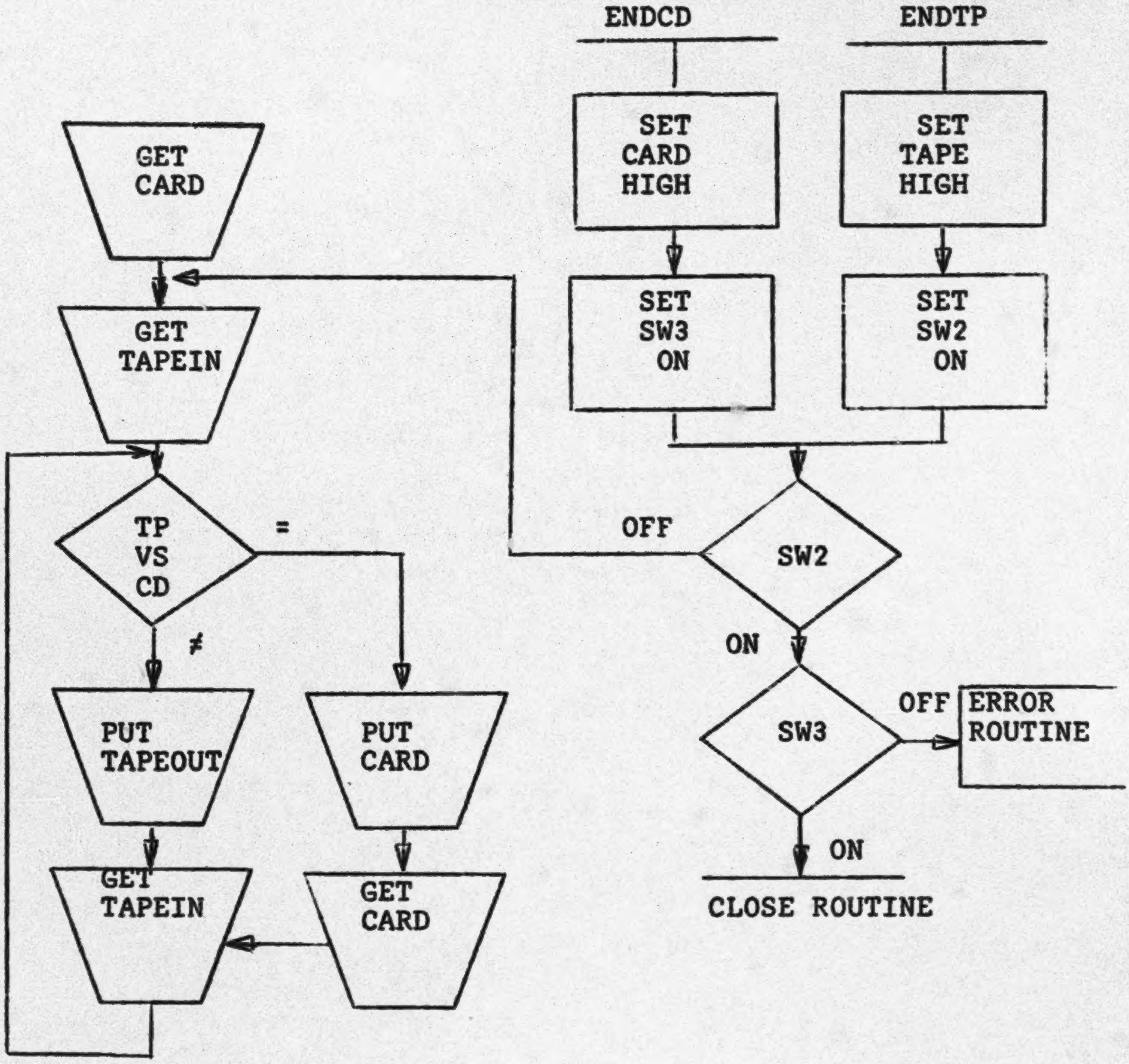
The Input Tape label has the following format:

```
// TPLAB INPUT PROBLEM 9A. 16KBOS00010001000000065356070365'
```

The IOMODS available in the relocatable library are as follows:

CDLOGIC	CDMOD	DEVICE=2540,RECFORM=FIXUNB, TYPEFLE=INPUT
PCHLOGIC	CDMOD	CRDERR=RETRY,DEVICE=2540,IOAREA2=YES, RECFORM=FIXUNB,TYPEFLE=OUTPUT,WORKA= YES
PRTLOGIC	PRMOD	CONTROL=YES,IOAREA2=YES,PRINTOV=YES, RECFORM=FIXUNB,WORKA=YES
TPLOGIC	MTMOD	RECFORM=FIXBLK,WORKA=YES,READ=FORWARD

Machine Problem 9A


ENDCD
ENDTP
GET CARD
SET CARD HIGH
SET TAPE HIGH
GET TAPEIN
SET SW3 ON
SET SW2 ON
TP VS CD
=
OFF
SW2
≠
ON
PUT TAPEOUT
PUT CARD
OFF
SW3
ERROR ROUTINE
ON
GET TAPEIN
GET CARD
CLOSE ROUTINE

```
* EXAMPLE
         TITLE 'SOLUTION FOR PROBLEM 9A.'
         REPRO
         PHASE PROB9A,S
BEGIN    START
TYPE     DTFCN BLKSIZE=49,DEVADDR=SYSLOG,IOAREA1=TYPE1,RECFORM=UNDEF,
               TYPEFLE=OUTPUT,WORKA=YES,RECSIZE=(4)
CARD     DTFCD BLKSIZE=30,DEVADDR=SYSIPT,DEVICE=2540,EOFADDR=ENDCD,
               IOAREA1=CDA1,MODNAME=CDLOGIC,RECFORM=FIXUNB,
               SSELECT=2,TYPEFLE=INPUT
PUNCH    DTFCD BLKSIZE=80,CRDERR=RETRY,DEVADDR=SYSPCH,DEVICE=2540,
               IOAREA1=PCH1,MODNAME=PCHLOGIC,RECFORM=FIXUNB,SSELECT=2,
               TYPEFLE=OUTPUT,IOAREA2=PCH2,WORKA=YES
LINE     DTFPR BLKSIZE=72,
               CONTROL=YES,
               DEVADDR=SYSLST,
               DEVICE=1403,
               MODNAME=PRTLOGIC,
               IOAREA1=PCH1,
               IOAREA2=PCH2,
               PRINTOV=YES,
               WORKA=YES
               RECFORM=FIXUNB
TAPEIN   DTFMT BLKSIZE=480,DEVADDR=SYS006,EOFADDR=ENDTP,FILABL=STD,
               IOAREA1=TPIN1,IOAREA2=TPIN2,MODNAME=TPLOGIC,
               RECFORM=FIXBLK,RECSIZE=80,TYPEFLE=INPUT,WORKA=YES
TAPEOUT  DTFMT BLKSIZE=960,DEVADDR=SYS004,FILABL=NO,IOAREA1=TPOUT1,
               IOAREA2=TPOUT2,MODNAME=TPLOGIC,RECFORM=FIXBLK,WORKA=YES,
               RECSIZE=80,TYPEFLE=OUTPUT
PRTAPE   DTFMT BLKSIZE=960,
               DEVADDR=SYS004,
               EOFADDR=EOJ,
               FILABL=NO,
               IOAREA1=TPOUT1,
               IOAREA2=TPOUT2,
               NODNAME=TPLOGIC,
               RECFORM=FIXBLK,
               WORKA=YES,
               RECSIZE=80,
               TYPEFLE=INPUT
PCH1     DS    CL80
PCH2     DS    CL80
TPIN1    DS    6CL80
TPIN2    DS    6CL80
TPOUT1   DS    12CL80
TPOUT2   DS    12CL80
TYPE1    DS    CL49
HERE     CSECT
         BALR  11,0
         USING *,11,10
BASE1    L     10,BASE2
         OPEN  CARD,PUNCH,TAPEIN,TAPEOUT
         BAL   9,GETCD
         BAL   9,GETTP
TPVSCD   CLC   CDA1+14(8),TPWRK+72
         BE    DELETE
         BAL   9,PUTAPE
GETAPE   BAL   9,GETTP
         B     TPVSCD
DELETE   BAL   9,PUTPCH
         BAL   9,GETCD
         B     GETAPE
```

```
ENDCD     MVC   CDA1+14(8),=X'FFFFFFFFFFFFFFFF'
          NI    SW3+1,X'OF'
SW2       B     TPVSCD
SW3       B     ERROR
          MVI   TPWRK,C' '
          MVC   TPWRK+1(79),TPWRK
          BAL   9,PUTPCH
          BAL   9,PUTPCH
          CLOSE CARD,PUNCH,TAPEIN,TAPEOUT
          CNTRL LINE,SK,1
          OPEN  LINE,PRTAPE
LOOP1     GET   PRTAPE,TPWRK
          PUT   LINE,TPWRK
          PRTOV LINE,12
          B     LOOP1
EOJ       LA    4,11
          PUT   TYPE,EOJMSG
          CNTRL LINE,SK,1
          CLOSE LINE,PRTAPE
          EOJ
ENDTP     MVC   TPWRK+72(8),=X'FFFFFFFFFFFFFFFF'
          NI    SW2+1,X'OF'
          B     SW2
GETCD     GET   CARD
          BR    9
GETTP     GET   TAPEIN,TPWRK
          BR    9
PUTTAPE   PUT   TAPEOUT,TPWRK
          BR    9
PUTPCH    PUT   PUNCH,TPWRK
          BR    9
ERROR     LA    4,49
          PUT   TYPE,ERRMSG
          DUMP
EOJMSG    DC    C'END OF JOB.'
ERRMSG    DC    C'INVALID DELETE CARD IN READER AT INPUT TAPE END.'
BASE2     DC    A(BASE1+4096)
CDA1      DS    CL30
TPWRK     DS    CL80
          LTORG
          END   HERE
```

MACHINE PROBLEM 9B

1. Request operator action from the console by displaying the message 'ENTER KEY' on the console.

2. Use the 'KEY' typed in by the operator to randomly retrieve a record from an Index Sequential file and display it on the console (SYSLOG).

3. Request operator action from the console by displaying the message 'ARE CORRECTIONS REQUIRED?' on the console.

4. Possible replies to this request are:

   a. NO
   b. YES

5. If a correction is not required determine if an additional inquiry is desired (Item #8).

6. If corrections are required display the message 'ENTER CORRECTIONS" on the console.

7. After corrections have been entered on the console:

   a. Update disk record.
   b. Write updated record back on disk.
   c. Retrieve updated record from disk.
   d. Display retrieved record on console for verification.

8. Request operator action from console by displaying the message 'REPEAT INQUIRY?'

9. Possible replies to this request are:

   a. NO
   b. YES

10. If no further inquiries are desired CLOSE the file.

11. If an additional inquiry is requested repeat entire inquiry routine beginning with the 'ENTER KEY' message.

12. If UPSI bit 5 is 'OFF' after closing file return control to the Supervisor (EOJ macro).

13. If UPSI bit 5 is 'ON' after closing branch to the contents of General Register 9.

14. Disk record format:

    a. Unblocked records.
    b. Key length 8 bytes.
    c. Data length 112 bytes.
    d. Corrections replace data record bytes 15-22.

15. Disk error messages:

    a. If machine error display 'DASD'.
    b. If wrong-length-record display 'WRONG LENGTH RECORD'.
    c. If end-of-file display 'END OF FILE'
    d. If no-record-found display 'NO RECORD FOUND'.
    e. If illegal ID display 'ILLEGAL ID SPECIFIED'.

16. Use symbolic unit SYS009 for the file.

17. The Cylinder Index will start on cylinder 197 track 0 of Physical drive number X'nnn'*. (1 cylinder)

18. The Prime Data Area will start on cylinder 198 track 0 of Physical Drive number X'nnn'*. (1 cylinder)

19. The name of the ISMOD to be used with this DTFIS is 'RTRVEMOD'.

20. The Label card used to create the Index Sequential file were as follows:

```
// DLAB   'DATA AREA FOR PROBLEM 9B IS RETRIEVAL PROB.  1111111',-
          0001,66001,66001,'16K DOS DISK',ISC
```

*nnn=contact instructor for this value.

```
* EXAMPLE
         TITLE 'SOLUTION FOR PROBLEM 98.'
         REPRO
         PHASE PROB9B,S
         START 0
CONSL    DTFCN BLKSIZE=112,
               DEVADDR=SYSLOG,
               IOAREA1=AREA3,
               RECFORM=UNDEF,
               RECSIZE=(10),
               TYPEFLE=INPUT,
               WORKA=YES
RANDM    DTFIS DSKXTNT=2,
               DEVICE=2311,
               HINDEX=2311,
               IOAREAR=AREA2,
               IOROUT=RETRVE,
               KEYARG=SEEKFLD,
               KEYLEN=8,
               MODNAME=RTRVEMOD,
               RECFORM=FIXUNB,
               NRECDS=1,
               RECSIZE=112,
               TYPEFLE=RANDOM,
               WORKR=DSKREC
RTRVEMOD ISMOD IOROUT=RETRVE,
               TYPEFLE=RANDOM
INTERUPT CSECT
         BALR  11,0
         USING *,11
         OPEN  RANDM
         L     8,=A(RANDMC)
LOOP     LA    10,11
         PUT   CONSL,MSG1
         GET   CONSL,AREA1
         MVC   SEEKFLD,AREA1
         READ  RANDM,KEY
         WAITF RANDM
         CLI   0(8),X'00'
         BNE   ERR1
         LA    10,112
         PUT   CONSL,DSKREC
         LA    10,28
         PUT   CONSL,MSG2
         GET   CONSL,AREA1
         OC    AREA1(3),=X'404040'
         CLC   AREA1(3),=C'YES'
         BNE   CLOSE
         LA    10,21
         PUT   CONSL,MSG3
         GET   CONSL,AREA1
         MVC   DSKREC+14(8),AREA1
         WRITE RANDM,KEY
         WAITF RANDM
         CLI   0(8),X'00'
         BNE   ERR1
         READ  RANDM,KEY
         WAITF RANDM
         CLI   0(8),X'00'
         BNE   ERR1
         LA    10,112
         PUT   CONSL,DSKREC
```

```
          LA     10,15
          PUT CONSL,MESG6
          GET    CONSL,AREA1
          OC     AREA1(3),=X'404040'
          CLC    AREA1(3),=C'YES'
          BE     LOOP
CLOSE     CLOSE RANDM
          COMRG
          TM     23(1),X'04'
          BZ     AROUND
          BR     9
AROUND    EOJ
ERR1      LA     10,20
          TM     0(8),X'80'
          BO     MSG4
          TM     0(8),X'40'
          PO     MSG5
          TM     0(8),X'20'
          BO     MSG6
          TM     0(8),X'10'
          BO     MSG7
          PUT    CONSL,MSG5
          B      CLOSE
MSG4      PUT    CONSL,MESG1
          B      CLOSE
MSG5      PUT    CONSL,MESG2
          B      CLOSE
MSG6      PUT    CONSL,MESG3
          B      CLOSE
MSG7      PUT    CONSL,MESG4
          B      CLOSE
MSG1      DC     C' ENTER KEY '
MSG2      DC     C'  ARE CORRECTIONS REQUIRED? '
MSG3      DC     C'  ENTER CORRECTIONS '
DSKREC    DS     CL112
AREA1     DC     CL112' '
AREA2     DC     CL130' '
AREA3     DC     CL130' '
SEEKFLD   DS     CL8
MESG1     DC     CL20'DASD'
MESG2     DC     CL20'WRONG LENGTH RECORD'
MESG3     DC     CL20'END OF FILE'
MESG4     DC     CL20'NO RECORD FOUND'
MESG5     DC     CL20'ILLEGAL ID SPECIFIED'
MESG6     DC     C'REPEAT INQUIRY?'
BIT       DC     X'04'
          LTORG
          END    INTERUPT
```

MACHINE PROBLEM 9C

1. Modify problem 9A to incorporate problem 9B (without alteration) into a single phase program by use of the proper External linkage symbols and control statements.

2. Add the STXIT to problem 9A to permit an operator to interrupt from the console (1052) and use the Random retrieval routine (problem 9B).

3. OPEN and Interrupt exit only if UPSI bit 5 is 'ON'.

```
* EXAMPLE
         TITLE 'SOLUTION FOR PROBLEM 9C.'
         REPRO
         PHASE PROB9C,S
BEGIN    START
TYPE     DTFCN BLKSIZE=49,DEVADDR=SYSLOG,IOAREA1=TYPE1,RECFORM=UNDEF,
               TYPEFLE=OUTPUT,WORKA=YES,RECSIZE=(4)
CARD     DTFCD BLKSIZE=30,DEVADDR=SYSIPT,DEVICE=2540,EOFADDR=ENDCD,
               IOAREA1=CDA1,MODNAME=CDLOGIC,RECFORM=FIXUNB,
               SSELECT=2,TYPEFLE=INPUT
PUNCH    DTFCD BLKSIZE=80,CRDERR=RETRY,DEVADDR=SYSPCH,DEVICE=2540,
               IOAREA1=PCH1,MODNAME=PCHLOGIC,RECFORM=FIXUNB,SSELECT=2.
               TYPEFLE=OUTPUT,IOAREA2=PCH2,WORKA=YES
LINE     DTFPR BLKSIZE=72,
               CONTROL=YES,
               DEVADDR=SYSLST,
               DEVICE=1403,
               MODNAME=PRTLOGIC,
               IOAREA1=PCH1,
               IOAREA2=PCH2,
               PRINTOV=YES,
               WORKA=YES,
               RECFORM=FIXUNB
TAPEIN   DTFMT BLKSIZE=480,DEVADDR=SYS006,EOFADDR=ENDTP,FILABL=STD,
               IOAREA1=TPIN1,IOAREA2=TPIN2,MODNAME=TPLOGIC,
               RECFORM=FIXBLK,RECSIZE=80,TYPEFLE=INPUT,WORKA=YES
TAPEOUT  DTFMT BLKSIZE=960,DEVADDR=SYS004,FILABL=NO,IOAREA1=TPOUT1,
               IOAREA2=TPOUT2,MODNAME=TPLOGIC,RECFORM=FIXBLK,WORKA=YES,
               RECSIZE=80,TYPEFLE=OUTPUT
PRTAPE   DTFMT BLKSIZE=960,
               DEVADDR=SYS004,
               EOFADDR=EOJ,
               FILABL=NO,
               IOAREA1=TPOUT1,
               IOAREA2=TPOUT2,
               MODNAME=TPLOGIC,
               RECFORM=FIXBLK,
               WORKA=YES,
               RECSIZE=80,
               TYPEFLE=INPUT
PCH1     DS    CL80
PCH2     DS    CL80
TPIN1    DS    6CL80
TPIN2    DS    6CL80
TPOUT1   DS    12CL80
TPOUT2   DS    12CL80
TYPE1    DS    CL49
HERE     CSECT
         BALR  11,0
         USING *,11,10
BASE1    L     10,BASE2
         B     COMRG
BASE2    DC    A(BASE1+4096)
COMRG    COMRG
         TM    23(1),X'04'
         BZ    OVER
         STXIT OC,SETUP,SAVE
OVER     OPEN  CARD,PUNCH,TAPEIN,TAPEOUT
         BAL   9,GETCD
         BAL   9,GETTP
TPVSCD   CLC   CDA1+14(8),TPWRK+72
         BE    DELETE
         BAL   9,PUTAPE
GETAPE   BAL   9,GETTP
         B     TPVSCD
DELETE   BAL   9,PUTPCH
         BAL   9,GETCD
         B     GETAPE
ENDCD    MVC   CDA1+14(8),=X'FFFFFFFFFFFFFFFF'
         NI    SW3+1,X'OF'
SW2      B     TPVSCD
SW3      B     ERROR
         MVI   TPWRK,C' '
         MVC   TPWRK+(79),TPWRK
         BAL   9,PUTPCH
         BAL   9,PUTPCH
         CLOSE CARD,PUNCH,TAPEIN,TAPEOUT
         CNTRL LINE,SK,1
REOPEN   OPEN  LINE,PRTAPE
LOOP1    GET   PRTAPE,TPWRK
         PUT   LINE,TPWRK
         PRTOV LINE,12
         B     LOOP1
EOJ      CNTRL LINE,SK,1
         CLOSE LINE,PRTAPE
         LA    4,11
         PUT   TYPE,EOJMSG
         EOJ
ENDTP    MVC   TPWRK+72(8),=X'FFFFFFFFFFFFFFF'
         NI    SW2+1,X'OF'
         B     SW2
GETCD    GET   CARD
         BR    9
GETTP    GET   TAPEIN,TPWRK
         BR    9
PUTAPE   PUT   TAPEOUT,TPWRK
         BR    9
PUTPCH   PUT   PUNCH,TPWRK
         BR    9
ERROR    LA    4,49
         PUT   TYPE,ERRMSG
         DUMP
SETUP    L     8,=V(INTERUPT)
         BALR  9,8
         EXIT  OC
SAVE     DS    OD
         DC    18F'O'
EOJMSG   DC    C'END OF JOB.'
ERRMSG   DC    C'INVALID DELETE CARD IN READER AT INPUT TAPE END.'
CDA1     DS    CL30
TPWRK    DS    CL80
         LTORG
         END   HERE
```